HEXAWARE

Assignment2

Task 1. Database Design:

1. Create the database named "SISDB".

2. Define the schema for the Students, Courses, Enrollments, Teacher, and Payments tables based on the provided schema. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

a. Students

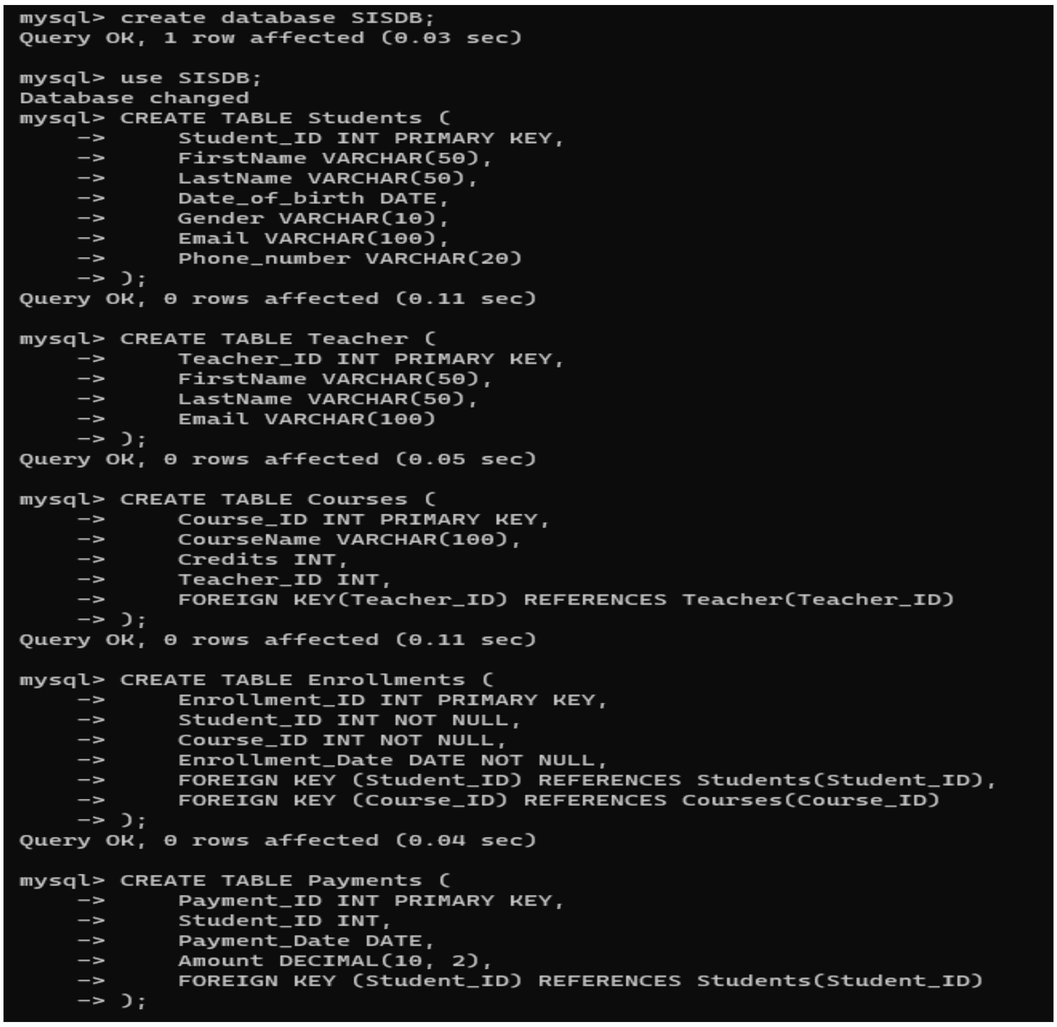
b. Courses

c. Enrollments

d. Teacher

e. Payments

4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.



```
mysql> create database SISDB;
Query OK, 1 row affected (0.03 sec)

mysql> use SISDB;
Database changed
mysql> CREATE TABLE Students (
    ->     Student_ID INT PRIMARY KEY,
    ->     FirstName VARCHAR(50),
    ->     LastName VARCHAR(50),
    ->     Date_of_birth DATE,
    ->     Gender VARCHAR(10),
    ->     Email VARCHAR(100),
    ->     Phone_number VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.11 sec)

mysql> CREATE TABLE Teacher (
    ->     Teacher_ID INT PRIMARY KEY,
    ->     FirstName VARCHAR(50),
    ->     LastName VARCHAR(50),
    ->     Email VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE Courses (
    ->     Course_ID INT PRIMARY KEY,
    ->     CourseName VARCHAR(100),
    ->     Credits INT,
    ->     Teacher_ID INT,
    ->     FOREIGN KEY(Teacher_ID) REFERENCES Teacher(Teacher_ID)
    -> );
Query OK, 0 rows affected (0.11 sec)

mysql> CREATE TABLE Enrollments (
    ->     Enrollment_ID INT PRIMARY KEY,
    ->     Student_ID INT NOT NULL,
    ->     Course_ID INT NOT NULL,
    ->     Enrollment_Date DATE NOT NULL,
    ->     FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID),
    ->     FOREIGN KEY (Course_ID) REFERENCES Courses(Course_ID)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Payments (
    ->     Payment_ID INT PRIMARY KEY,
    ->     Student_ID INT,
    ->     Payment_Date DATE,
    ->     Amount DECIMAL(10, 2),
    ->     FOREIGN KEY (Student_ID) REFERENCES Students(Student_ID)
    -> );
```

5. Insert at least 10 sample records into each of the following tables.

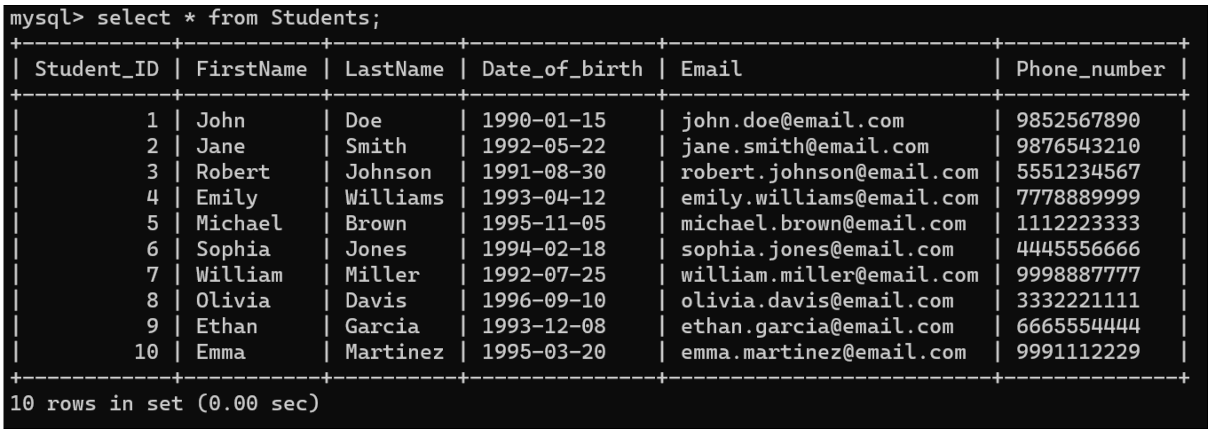
i. Students

ii. Courses

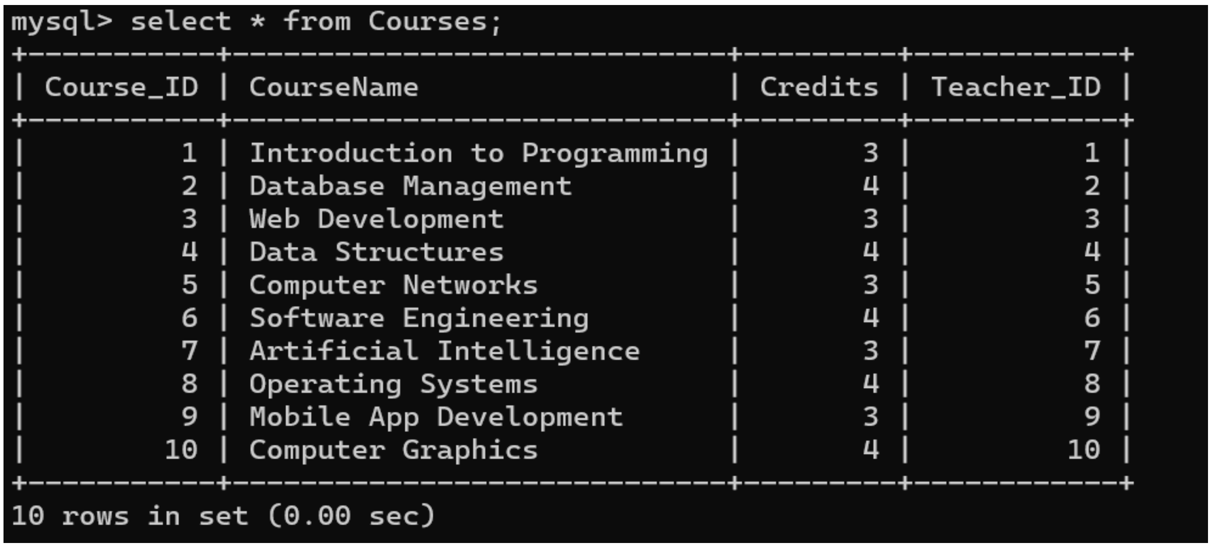
ill. Enrollments

iv. Teacher

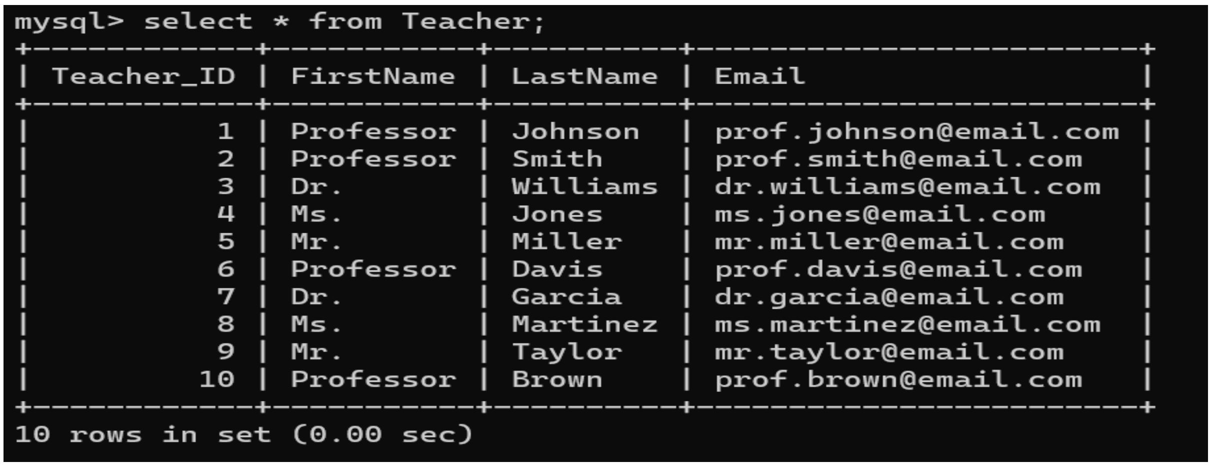
v. Payments



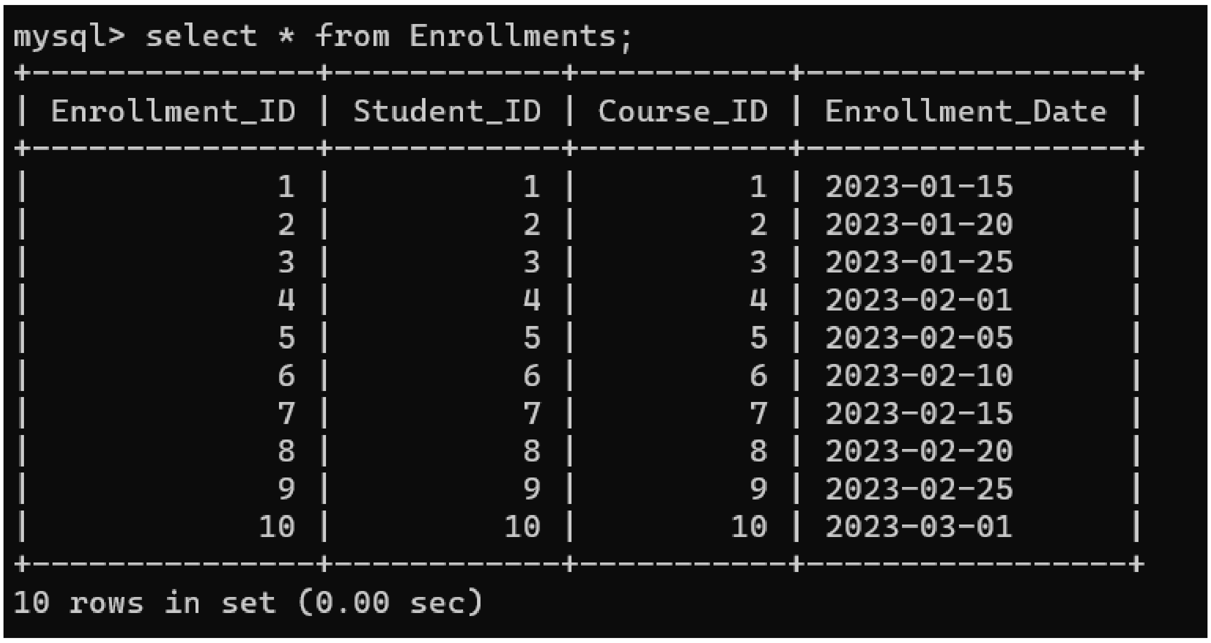
```
mysql> select * from Students;
+------------+-----------+----------+---------------+--------------------------+--------------+
| Student_ID | FirstName | LastName | Date_of_birth | Email                    | Phone_number |
+------------+-----------+----------+---------------+--------------------------+--------------+
|          1 | John      | Doe      | 1990-01-15    | john.doe@email.com       | 9852567890   |
|          2 | Jane      | Smith    | 1992-05-22    | jane.smith@email.com     | 9876543210   |
|          3 | Robert    | Johnson  | 1991-08-30    | robert.johnson@email.com | 5551234567   |
|          4 | Emily     | Williams | 1993-04-12    | emily.williams@email.com | 7778889999   |
|          5 | Michael   | Brown    | 1995-11-05    | michael.brown@email.com  | 1112223333   |
|          6 | Sophia    | Jones    | 1994-02-18    | sophia.jones@email.com   | 4445556666   |
|          7 | William   | Miller   | 1992-07-25    | william.miller@email.com | 9998887777   |
|          8 | Olivia    | Davis    | 1996-09-10    | olivia.davis@email.com   | 3332221111   |
|          9 | Ethan     | Garcia   | 1993-12-08    | ethan.garcia@email.com   | 6665554444   |
|         10 | Emma      | Martinez | 1995-03-20    | emma.martinez@email.com  | 9991112229   |
+------------+-----------+----------+---------------+--------------------------+--------------+
10 rows in set (0.00 sec)
```



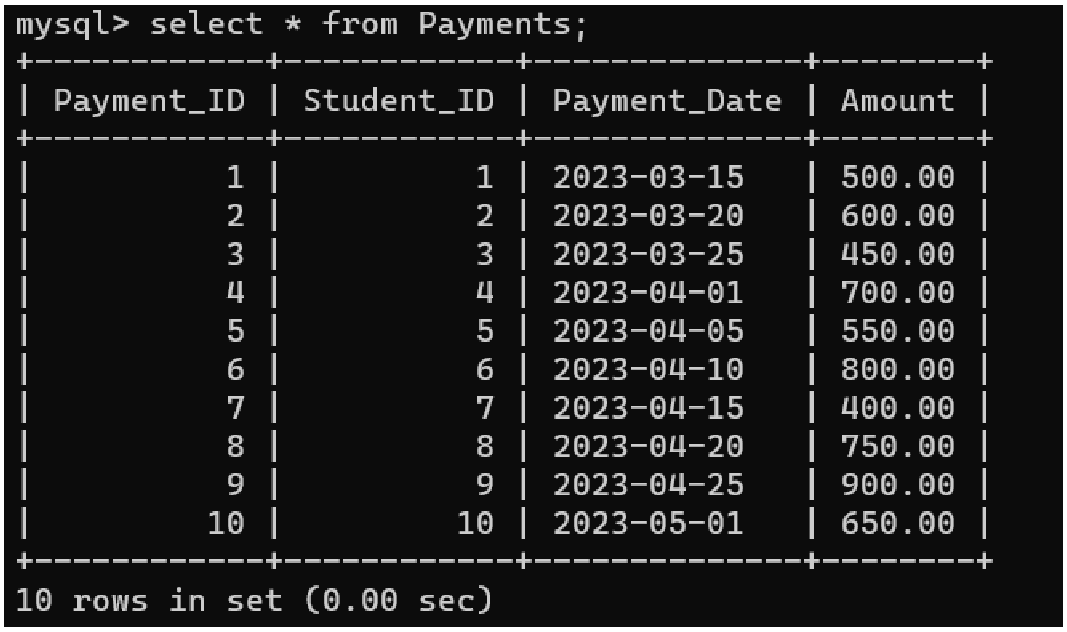
```
mysql> select * from Courses;
+-----------+-----------------------------+---------+------------+
| Course_ID | CourseName                  | Credits | Teacher_ID |
+-----------+-----------------------------+---------+------------+
|         1 | Introduction to Programming |       3 |          1 |
|         2 | Database Management         |       4 |          2 |
|         3 | Web Development             |       3 |          3 |
|         4 | Data Structures             |       4 |          4 |
|         5 | Computer Networks           |       3 |          5 |
|         6 | Software Engineering        |       4 |          6 |
|         7 | Artificial Intelligence     |       3 |          7 |
|         8 | Operating Systems           |       4 |          8 |
|         9 | Mobile App Development      |       3 |          9 |
|        10 | Computer Graphics           |       4 |         10 |
+-----------+-----------------------------+---------+------------+
10 rows in set (0.00 sec)
```



```
mysql> select * from Teacher;
+------------+-----------+----------+-------------------------+
| Teacher_ID | FirstName | LastName | Email                   |
+------------+-----------+----------+-------------------------+
|          1 | Professor | Johnson  | prof.johnson@email.com  |
|          2 | Professor | Smith    | prof.smith@email.com    |
|          3 | Dr.       | Williams | dr.williams@email.com   |
|          4 | Ms.       | Jones    | ms.jones@email.com      |
|          5 | Mr.       | Miller   | mr.miller@email.com     |
|          6 | Professor | Davis    | prof.davis@email.com    |
|          7 | Dr.       | Garcia   | dr.garcia@email.com     |
|          8 | Ms.       | Martinez | ms.martinez@email.com   |
|          9 | Mr.       | Taylor   | mr.taylor@email.com     |
|         10 | Professor | Brown    | prof.brown@email.com    |
+------------+-----------+----------+-------------------------+
10 rows in set (0.00 sec)
```



```
mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| Enrollment_ID | Student_ID | Course_ID | Enrollment_Date |
+---------------+------------+-----------+-----------------+
|             1 |          1 |         1 | 2023-01-15      |
|             2 |          2 |         2 | 2023-01-20      |
|             3 |          3 |         3 | 2023-01-25      |
|             4 |          4 |         4 | 2023-02-01      |
|             5 |          5 |         5 | 2023-02-05      |
|             6 |          6 |         6 | 2023-02-10      |
|             7 |          7 |         7 | 2023-02-15      |
|             8 |          8 |         8 | 2023-02-20      |
|             9 |          9 |         9 | 2023-02-25      |
|            10 |         10 |        10 | 2023-03-01      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```



```
mysql> select * from Payments;
+------------+------------+--------------+--------+
| Payment_ID | Student_ID | Payment_Date | Amount |
+------------+------------+--------------+--------+
|          1 |          1 | 2023-03-15   | 500.00 |
|          2 |          2 | 2023-03-20   | 600.00 |
|          3 |          3 | 2023-03-25   | 450.00 |
|          4 |          4 | 2023-04-01   | 700.00 |
|          5 |          5 | 2023-04-05   | 550.00 |
|          6 |          6 | 2023-04-10   | 800.00 |
|          7 |          7 | 2023-04-15   | 400.00 |
|          8 |          8 | 2023-04-20   | 750.00 |
|          9 |          9 | 2023-04-25   | 900.00 |
|         10 |         10 | 2023-05-01   | 650.00 |
+------------+------------+--------------+--------+
10 rows in set (0.00 sec)
```

Tasks 2: Select, Where, Between, AND, LIKE:

1. Write an SQL query to insert a new student into the "Students" table with the following details:

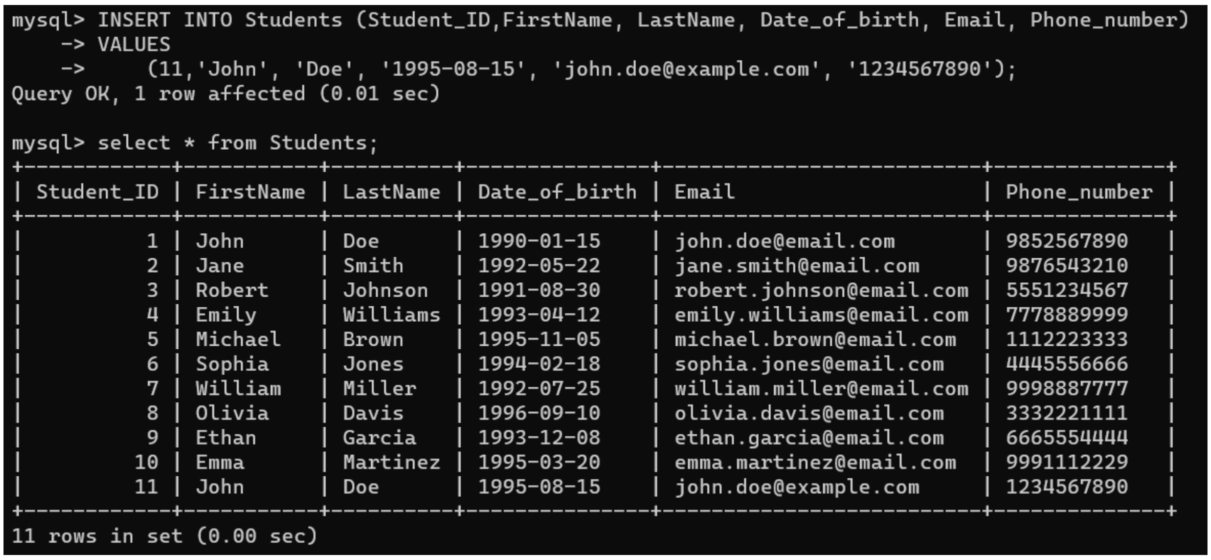
a. First Name: John

b. Last Name: Doe

c. Date of Birth: 1995-08-15

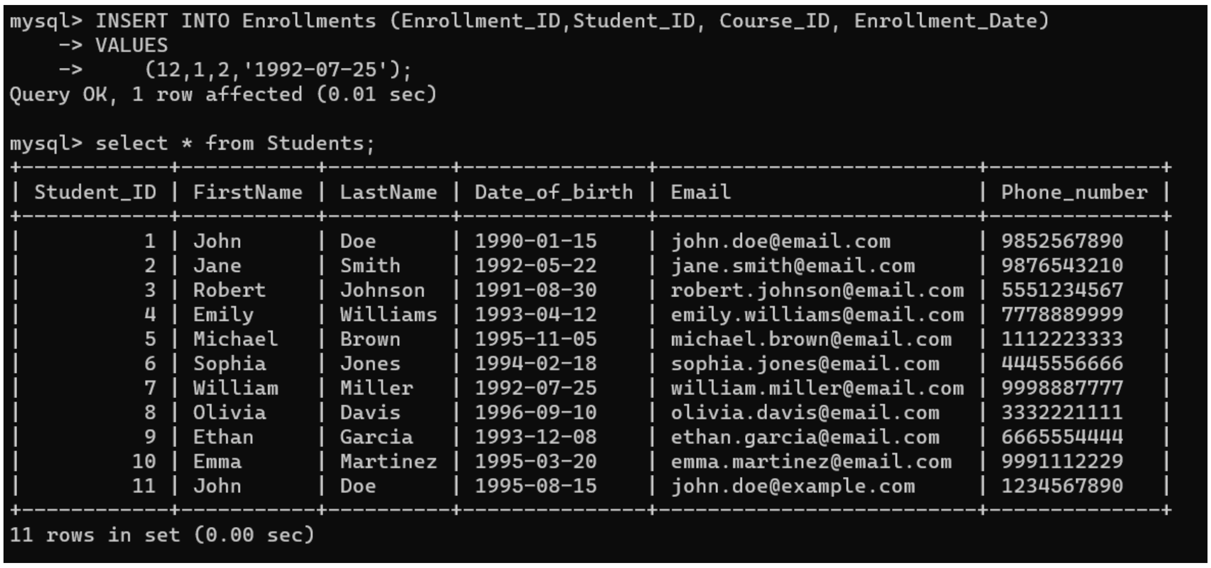
d. Email: john.doe@example.com

e. Phone Number: 1234567890



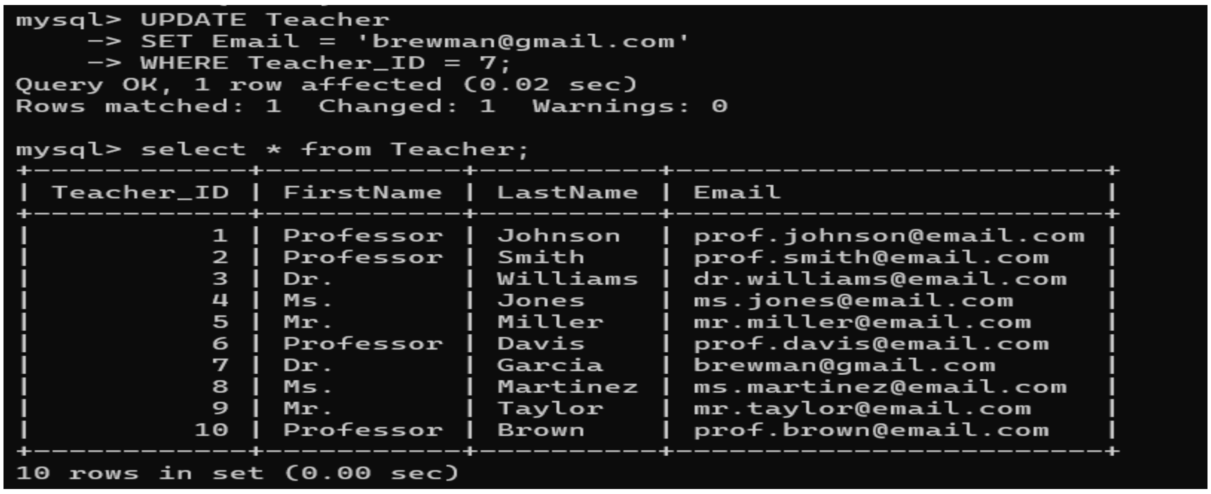
```
mysql> INSERT INTO Students (Student_ID,FirstName, LastName, Date_of_birth, Email, Phone_number)
    -> VALUES
    ->     (11,'John', 'Doe', '1995-08-15', 'john.doe@example.com', '1234567890');
Query OK, 1 row affected (0.01 sec)

mysql> select * from Students;
+------------+-----------+----------+---------------+---------------------------+--------------+
| Student_ID | FirstName | LastName | Date_of_birth | Email                     | Phone_number |
+------------+-----------+----------+---------------+---------------------------+--------------+
|          1 | John      | Doe      | 1990-01-15    | john.doe@email.com        | 9852567890   |
|          2 | Jane      | Smith    | 1992-05-22    | jane.smith@email.com      | 9876543210   |
|          3 | Robert    | Johnson  | 1991-08-30    | robert.johnson@email.com  | 5551234567   |
|          4 | Emily     | Williams | 1993-04-12    | emily.williams@email.com  | 7778889999   |
|          5 | Michael   | Brown    | 1995-11-05    | michael.brown@email.com   | 1112223333   |
|          6 | Sophia    | Jones    | 1994-02-18    | sophia.jones@email.com    | 4445556666   |
|          7 | William   | Miller   | 1992-07-25    | william.miller@email.com  | 9998887777   |
|          8 | Olivia    | Davis    | 1996-09-10    | olivia.davis@email.com    | 3332221111   |
|          9 | Ethan     | Garcia   | 1993-12-08    | ethan.garcia@email.com    | 6665554444   |
|         10 | Emma      | Martinez | 1995-03-20    | emma.martinez@email.com   | 9991112229   |
|         11 | John      | Doe      | 1995-08-15    | john.doe@example.com      | 1234567890   |
+------------+-----------+----------+---------------+---------------------------+--------------+
11 rows in set (0.00 sec)
```

2. Write an SQL query to enroll a student in a course. Choose an existing student and course and insert a record into the "Enrollments" table with the enrollment date.



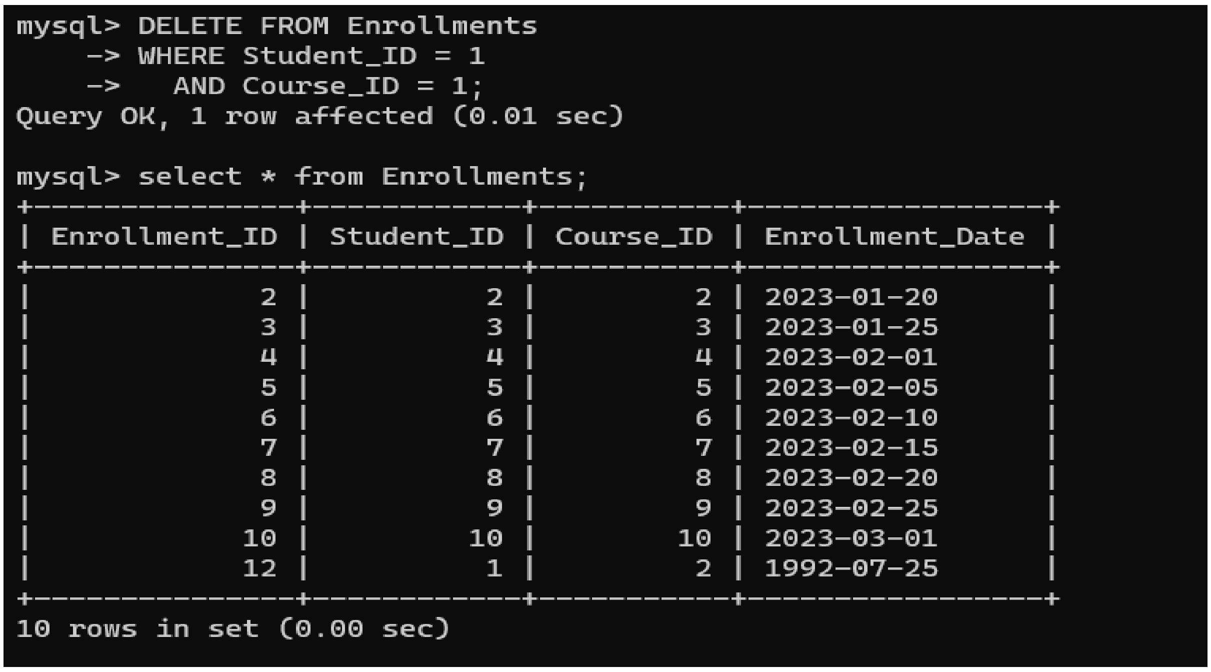
```
mysql> INSERT INTO Enrollments (Enrollment_ID,Student_ID, Course_ID, Enrollment_Date)
    -> VALUES
    ->     (12,1,2,'1992-07-25');
Query OK, 1 row affected (0.01 sec)

mysql> select * from Students;
+------------+-----------+----------+---------------+---------------------------+--------------+
| Student_ID | FirstName | LastName | Date_of_birth | Email                     | Phone_number |
+------------+-----------+----------+---------------+---------------------------+--------------+
|          1 | John      | Doe      | 1990-01-15    | john.doe@email.com        | 9852567890   |
|          2 | Jane      | Smith    | 1992-05-22    | jane.smith@email.com      | 9876543210   |
|          3 | Robert    | Johnson  | 1991-08-30    | robert.johnson@email.com  | 5551234567   |
|          4 | Emily     | Williams | 1993-04-12    | emily.williams@email.com  | 7778889999   |
|          5 | Michael   | Brown    | 1995-11-05    | michael.brown@email.com   | 1112223333   |
|          6 | Sophia    | Jones    | 1994-02-18    | sophia.jones@email.com    | 4445556666   |
|          7 | William   | Miller   | 1992-07-25    | william.miller@email.com  | 9998887777   |
|          8 | Olivia    | Davis    | 1996-09-10    | olivia.davis@email.com    | 3332221111   |
|          9 | Ethan     | Garcia   | 1993-12-08    | ethan.garcia@email.com    | 6665554444   |
|         10 | Emma      | Martinez | 1995-03-20    | emma.martinez@email.com   | 9991112229   |
|         11 | John      | Doe      | 1995-08-15    | john.doe@example.com      | 1234567890   |
+------------+-----------+----------+---------------+---------------------------+--------------+
11 rows in set (0.00 sec)
```

3. Update the email address of a specific teacher in the "Teacher" table. Choose any teacher and modify their email address.



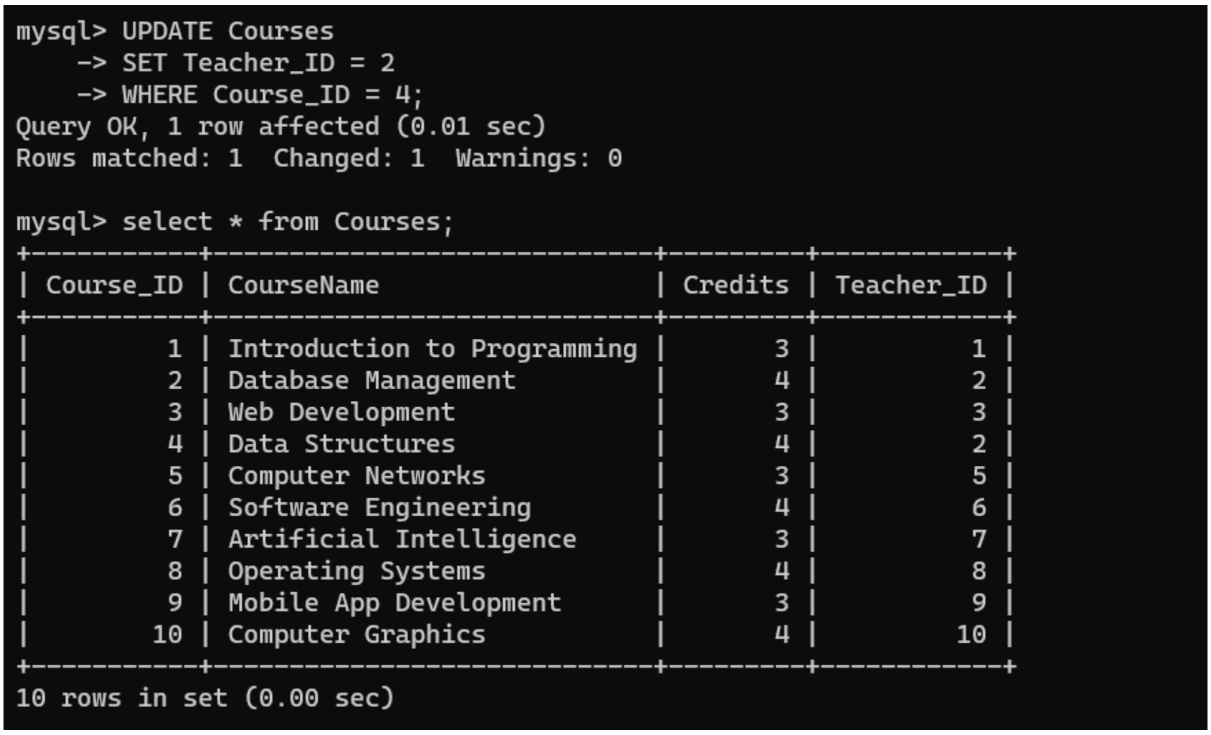
```
mysql> UPDATE Teacher
    -> SET Email = 'brewman@gmail.com'
    -> WHERE Teacher_ID = 7;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Teacher;
+------------+-----------+----------+-------------------------+
| Teacher_ID | FirstName | LastName | Email                   |
+------------+-----------+----------+-------------------------+
|          1 | Professor | Johnson  | prof.johnson@email.com  |
|          2 | Professor | Smith    | prof.smith@email.com    |
|          3 | Dr.       | Williams | dr.williams@email.com   |
|          4 | Ms.       | Jones    | ms.jones@email.com      |
|          5 | Mr.       | Miller   | mr.miller@email.com     |
|          6 | Professor | Davis    | prof.davis@email.com    |
|          7 | Dr.       | Garcia   | brewman@gmail.com       |
|          8 | Ms.       | Martinez | ms.martinez@email.com   |
|          9 | Mr.       | Taylor   | mr.taylor@email.com     |
|         10 | Professor | Brown    | prof.brown@email.com    |
+------------+-----------+----------+-------------------------+
10 rows in set (0.00 sec)
```

4. Write an SQL query to delete a specific enrollment record from the "Enrollments" table. Select an enrollment record based on the student and course.



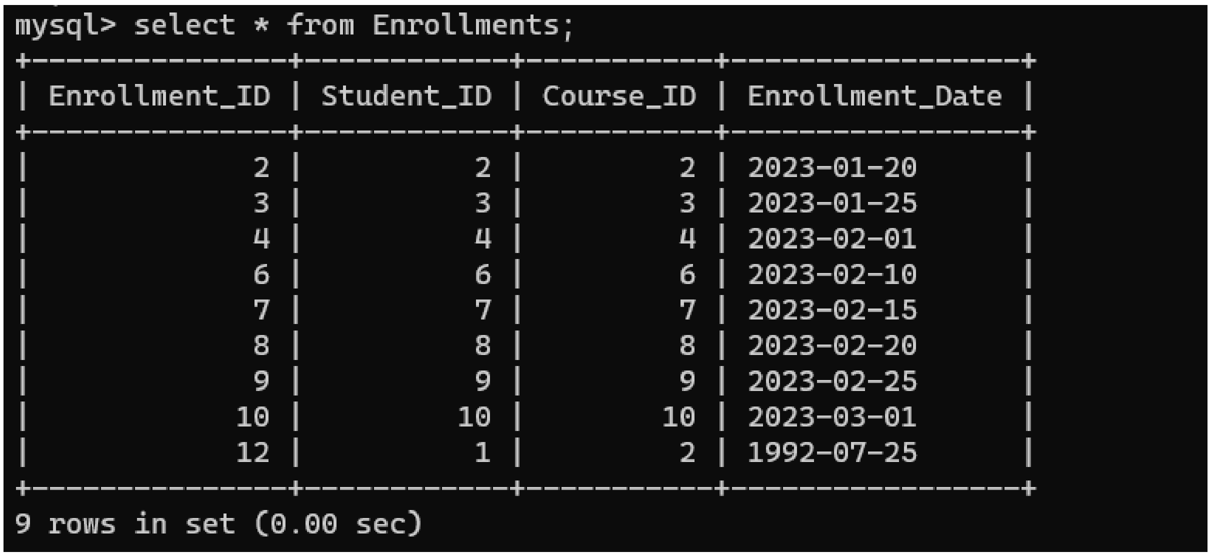
```
mysql> DELETE FROM Enrollments
    -> WHERE Student_ID = 1
    ->   AND Course_ID = 1;
Query OK, 1 row affected (0.01 sec)

mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| Enrollment_ID | Student_ID | Course_ID | Enrollment_Date |
+---------------+------------+-----------+-----------------+
|             2 |          2 |         2 | 2023-01-20      |
|             3 |          3 |         3 | 2023-01-25      |
|             4 |          4 |         4 | 2023-02-01      |
|             5 |          5 |         5 | 2023-02-05      |
|             6 |          6 |         6 | 2023-02-10      |
|             7 |          7 |         7 | 2023-02-15      |
|             8 |          8 |         8 | 2023-02-20      |
|             9 |          9 |         9 | 2023-02-25      |
|            10 |         10 |        10 | 2023-03-01      |
|            12 |          1 |         2 | 1992-07-25      |
+---------------+------------+-----------+-----------------+
10 rows in set (0.00 sec)
```

5. Update the "Courses" table to assign a specific teacher to a course. Choose any course and teacher from the respective tables.



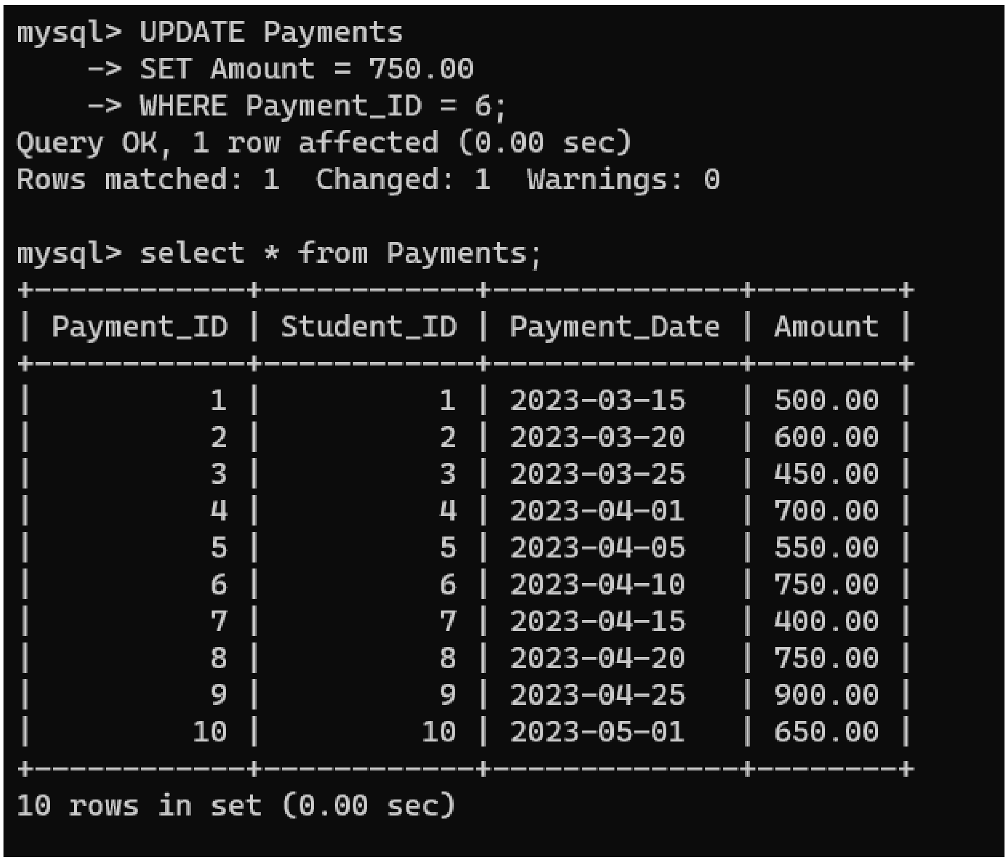
```
mysql> UPDATE Courses
    -> SET Teacher_ID = 2
    -> WHERE Course_ID = 4;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Courses;
+-----------+-----------------------------+---------+------------+
| Course_ID | CourseName                  | Credits | Teacher_ID |
+-----------+-----------------------------+---------+------------+
|         1 | Introduction to Programming |       3 |          1 |
|         2 | Database Management         |       4 |          2 |
|         3 | Web Development             |       3 |          3 |
|         4 | Data Structures             |       4 |          2 |
|         5 | Computer Networks           |       3 |          5 |
|         6 | Software Engineering        |       4 |          6 |
|         7 | Artificial Intelligence     |       3 |          7 |
|         8 | Operating Systems           |       4 |          8 |
|         9 | Mobile App Development      |       3 |          9 |
|        10 | Computer Graphics           |       4 |         10 |
+-----------+-----------------------------+---------+------------+
10 rows in set (0.00 sec)
```

6. Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.



```
mysql> select * from Enrollments;
+---------------+------------+-----------+-----------------+
| Enrollment_ID | Student_ID | Course_ID | Enrollment_Date |
+---------------+------------+-----------+-----------------+
|             2 |          2 |         2 | 2023-01-20      |
|             3 |          3 |         3 | 2023-01-25      |
|             4 |          4 |         4 | 2023-02-01      |
|             6 |          6 |         6 | 2023-02-10      |
|             7 |          7 |         7 | 2023-02-15      |
|             8 |          8 |         8 | 2023-02-20      |
|             9 |          9 |         9 | 2023-02-25      |
|            10 |         10 |        10 | 2023-03-01      |
|            12 |          1 |         2 | 1992-07-25      |
+---------------+------------+-----------+-----------------+
9 rows in set (0.00 sec)
```

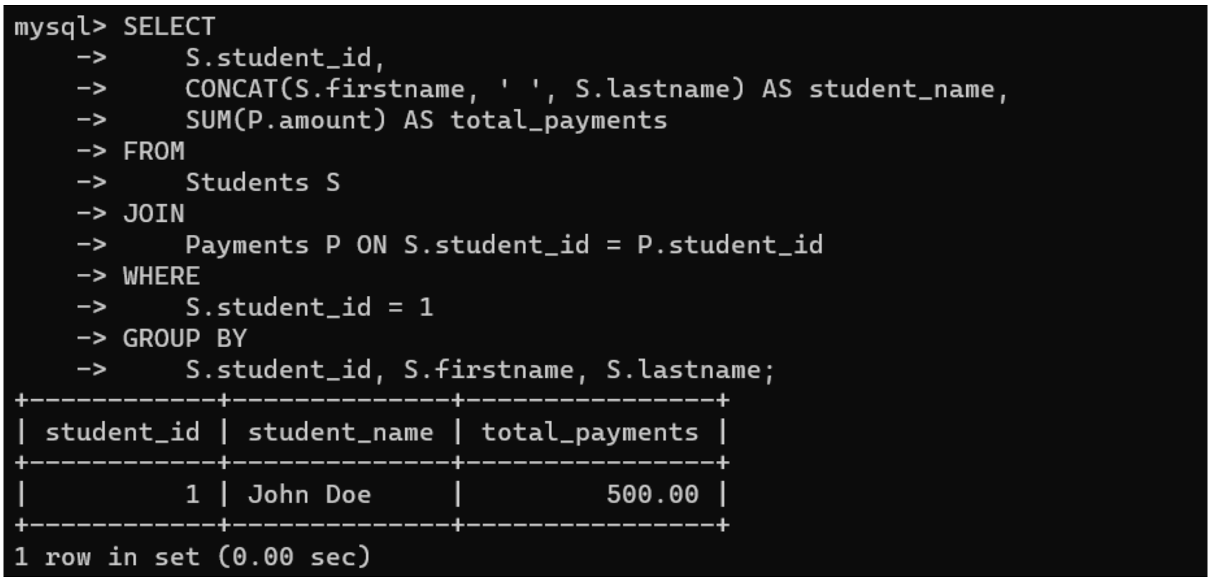
7. Update the payment amount for a specific payment record in the "Payments" table. Choose any payment record and modify the payment amount.



```
mysql> UPDATE Payments
    -> SET Amount = 750.00
    -> WHERE Payment_ID = 6;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Payments;
+------------+------------+--------------+--------+
| Payment_ID | Student_ID | Payment_Date | Amount |
+------------+------------+--------------+--------+
|          1 |          1 | 2023-03-15   | 500.00 |
|          2 |          2 | 2023-03-20   | 600.00 |
|          3 |          3 | 2023-03-25   | 450.00 |
|          4 |          4 | 2023-04-01   | 700.00 |
|          5 |          5 | 2023-04-05   | 550.00 |
|          6 |          6 | 2023-04-10   | 750.00 |
|          7 |          7 | 2023-04-15   | 400.00 |
|          8 |          8 | 2023-04-20   | 750.00 |
|          9 |          9 | 2023-04-25   | 900.00 |
|         10 |         10 | 2023-05-01   | 650.00 |
+------------+------------+--------------+--------+
10 rows in set (0.00 sec)
```

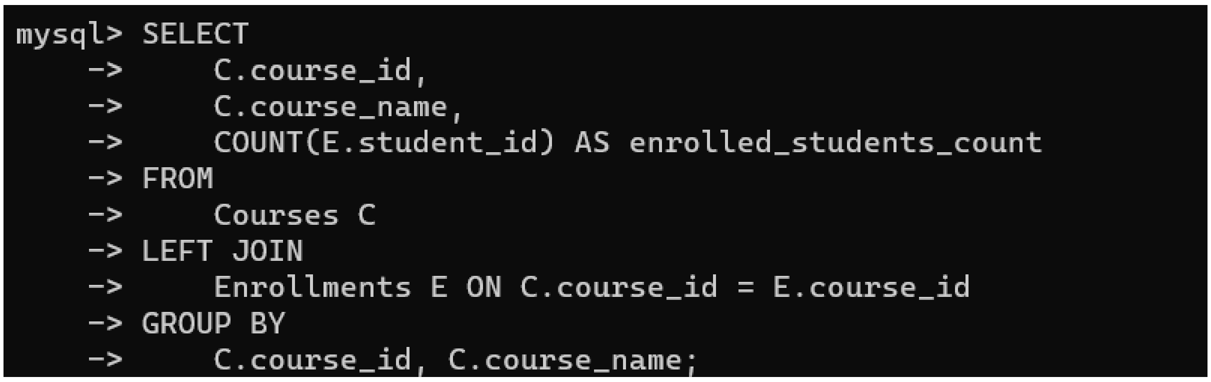
Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

1. Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.



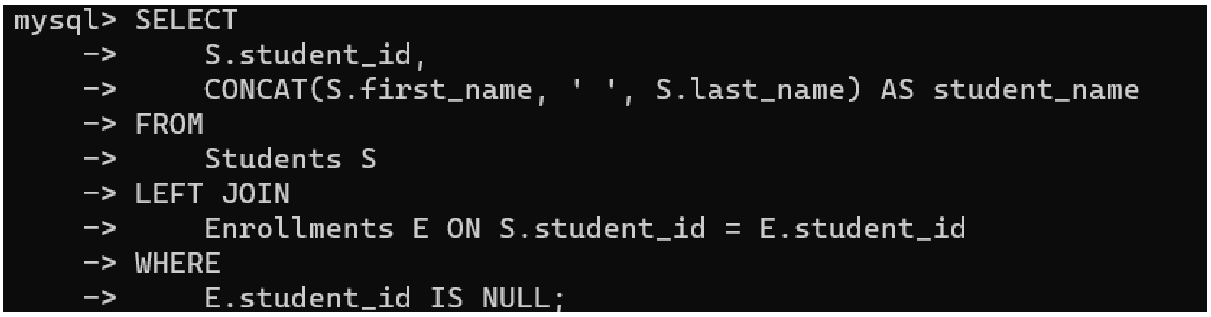
```
mysql> SELECT
    ->     S.student_id,
    ->     CONCAT(S.firstname, ' ', S.lastname) AS student_name,
    ->     SUM(P.amount) AS total_payments
    -> FROM
    ->     Students S
    -> JOIN
    ->     Payments P ON S.student_id = P.student_id
    -> WHERE
    ->     S.student_id = 1
    -> GROUP BY
    ->     S.student_id, S.firstname, S.lastname;
+------------+--------------+----------------+
| student_id | student_name | total_payments |
+------------+--------------+----------------+
|          1 | John Doe     |         500.00 |
+------------+--------------+----------------+
1 row in set (0.00 sec)
```

2. Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.



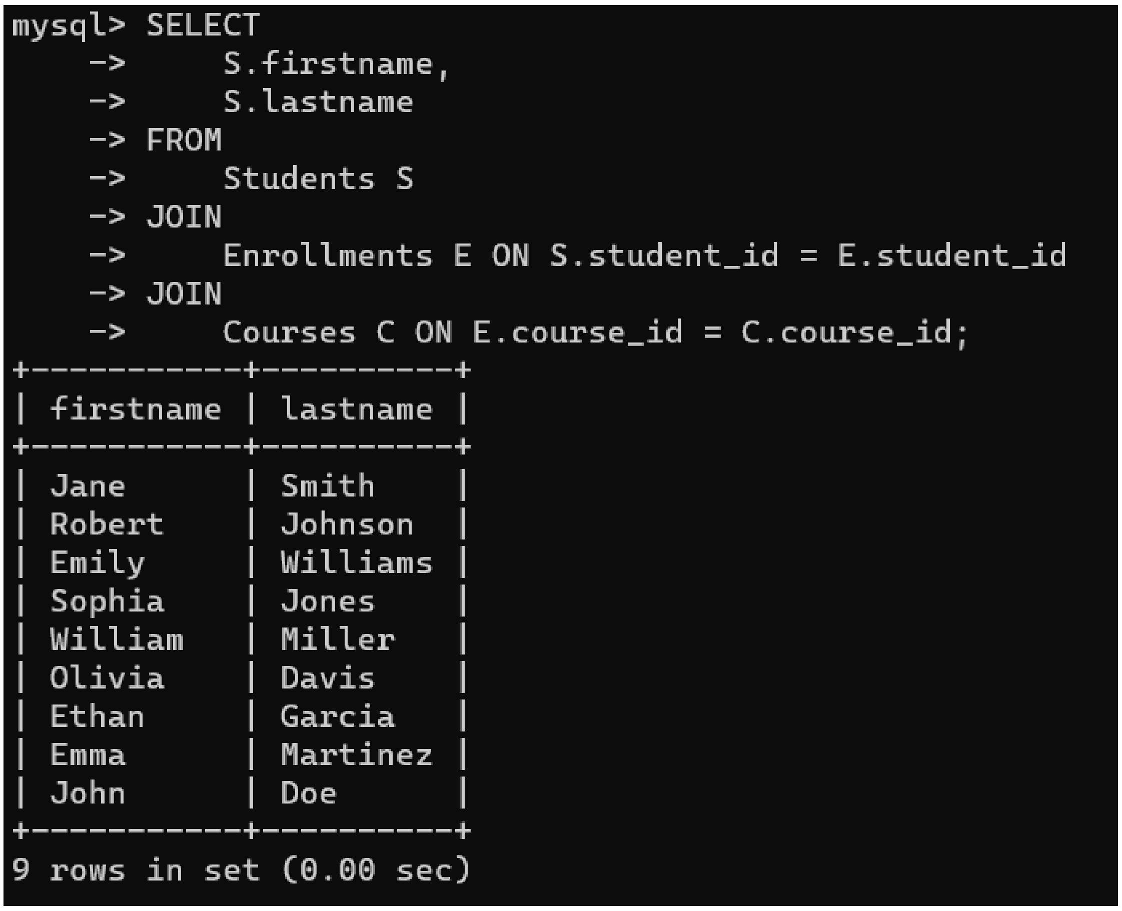
```
mysql> SELECT
    ->     C.course_id,
    ->     C.course_name,
    ->     COUNT(E.student_id) AS enrolled_students_count
    -> FROM
    ->     Courses C
    -> LEFT JOIN
    ->     Enrollments E ON C.course_id = E.course_id
    -> GROUP BY
    ->     C.course_id, C.course_name;
```

3. Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments.



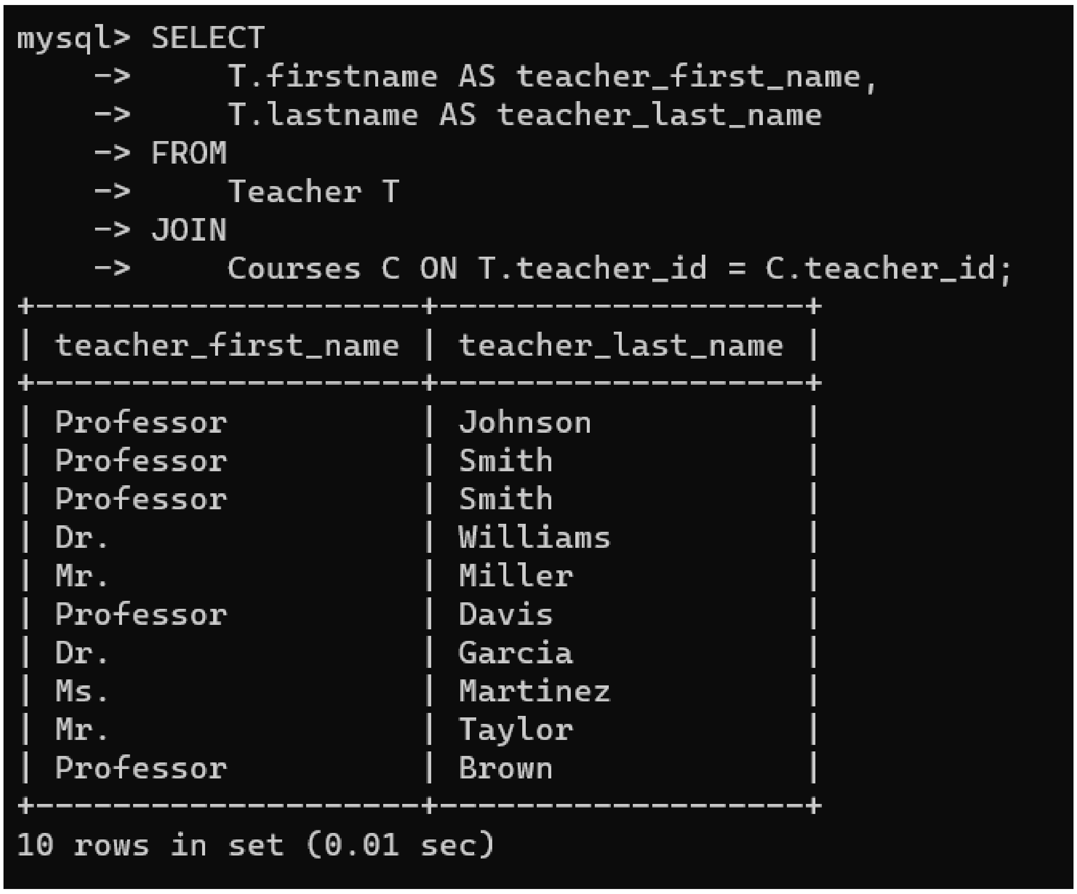
```
mysql> SELECT
    ->     S.student_id,
    ->     CONCAT(S.first_name, ' ', S.last_name) AS student_name
    -> FROM
    ->     Students S
    -> LEFT JOIN
    ->     Enrollments E ON S.student_id = E.student_id
    -> WHERE
    ->     E.student_id IS NULL;
```

4. Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.



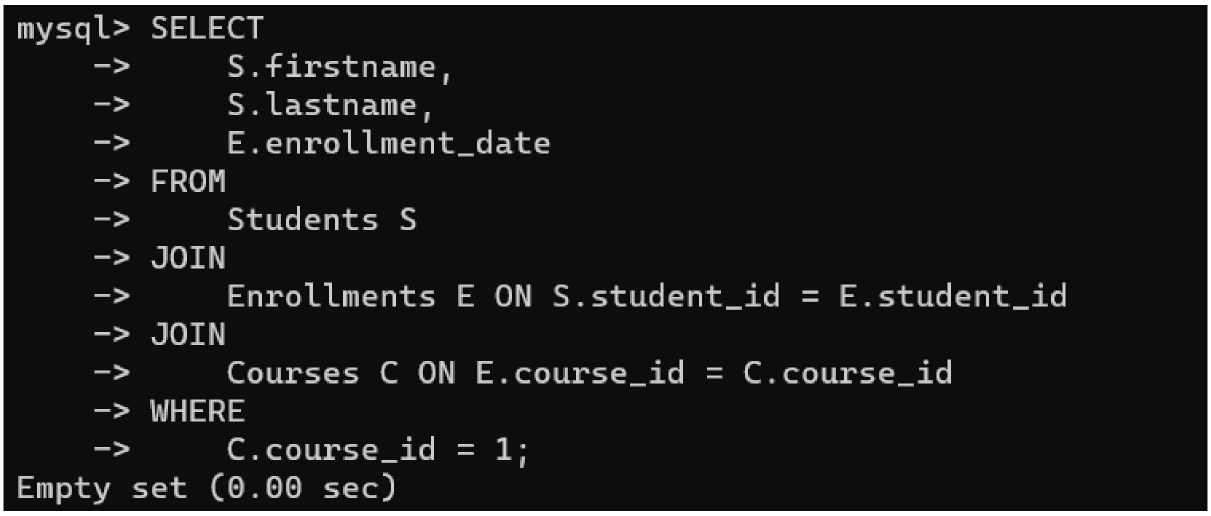
```
mysql> SELECT
    ->     S.firstname,
    ->     S.lastname
    -> FROM
    ->     Students S
    -> JOIN
    ->     Enrollments E ON S.student_id = E.student_id
    -> JOIN
    ->     Courses C ON E.course_id = C.course_id;
+-----------+----------+
| firstname | lastname |
+-----------+----------+
| Jane      | Smith    |
| Robert    | Johnson  |
| Emily     | Williams |
| Sophia    | Jones    |
| William   | Miller   |
| Olivia    | Davis    |
| Ethan     | Garcia   |
| Emma      | Martinez |
| John      | Doe      |
+-----------+----------+
9 rows in set (0.00 sec)
```

5. Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.



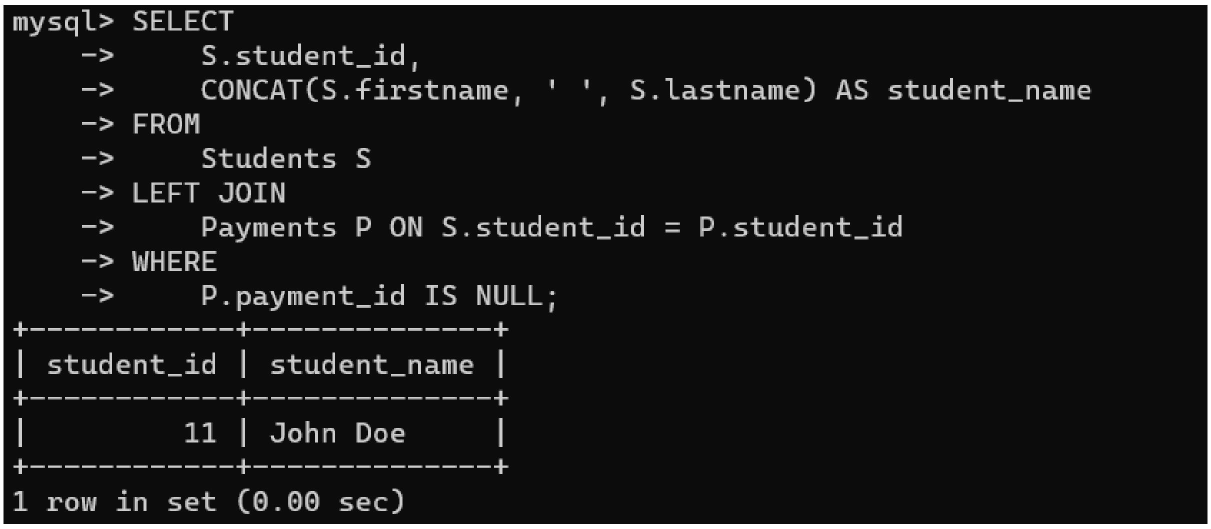
```
mysql> SELECT
    ->     T.firstname AS teacher_first_name,
    ->     T.lastname AS teacher_last_name
    -> FROM
    ->     Teacher T
    -> JOIN
    ->     Courses C ON T.teacher_id = C.teacher_id;
+--------------------+-------------------+
| teacher_first_name | teacher_last_name |
+--------------------+-------------------+
| Professor          | Johnson           |
| Professor          | Smith             |
| Professor          | Smith             |
| Dr.                | Williams          |
| Mr.                | Miller            |
| Professor          | Davis             |
| Dr.                | Garcia            |
| Ms.                | Martinez          |
| Mr.                | Taylor            |
| Professor          | Brown             |
+--------------------+-------------------+
10 rows in set (0.01 sec)
```

6. Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.



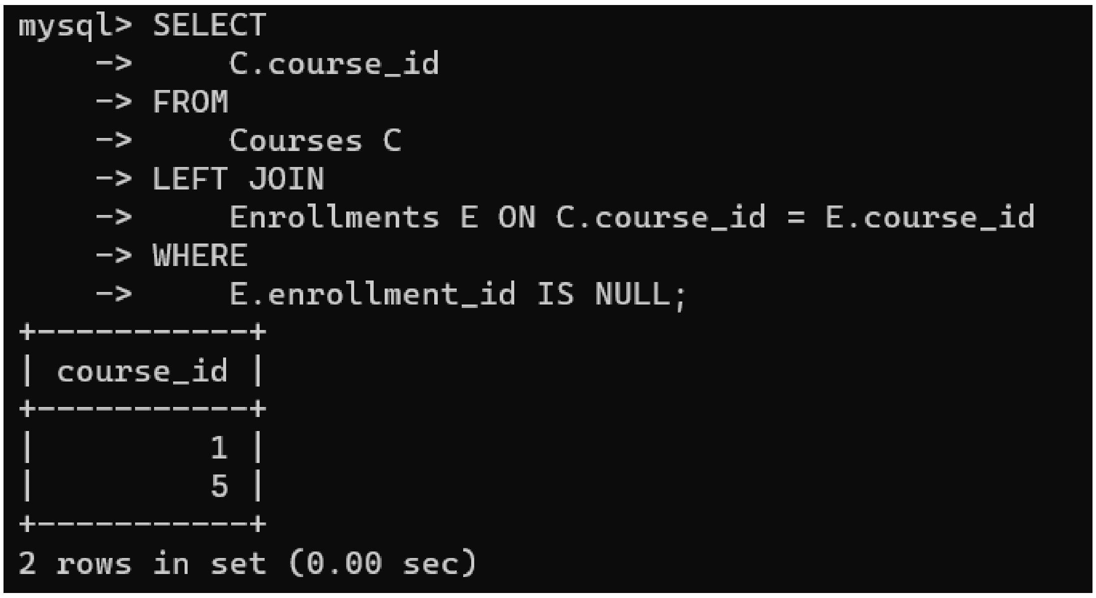
```
mysql> SELECT
    ->     S.firstname,
    ->     S.lastname,
    ->     E.enrollment_date
    -> FROM
    ->     Students S
    -> JOIN
    ->     Enrollments E ON S.student_id = E.student_id
    -> JOIN
    ->     Courses C ON E.course_id = C.course_id
    -> WHERE
    ->     C.course_id = 1;
Empty set (0.00 sec)
```

7. Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records.



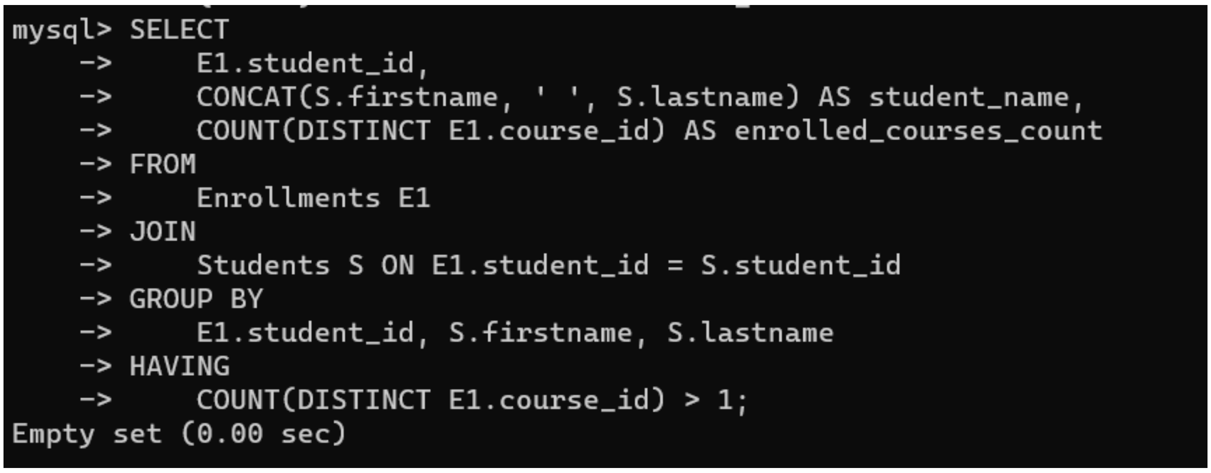
```
mysql> SELECT
    ->     S.student_id,
    ->     CONCAT(S.firstname, ' ', S.lastname) AS student_name
    -> FROM
    ->     Students S
    -> LEFT JOIN
    ->     Payments P ON S.student_id = P.student_id
    -> WHERE
    ->     P.payment_id IS NULL;
+------------+--------------+
| student_id | student_name |
+------------+--------------+
|         11 | John Doe     |
+------------+--------------+
1 row in set (0.00 sec)
```

8. Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.



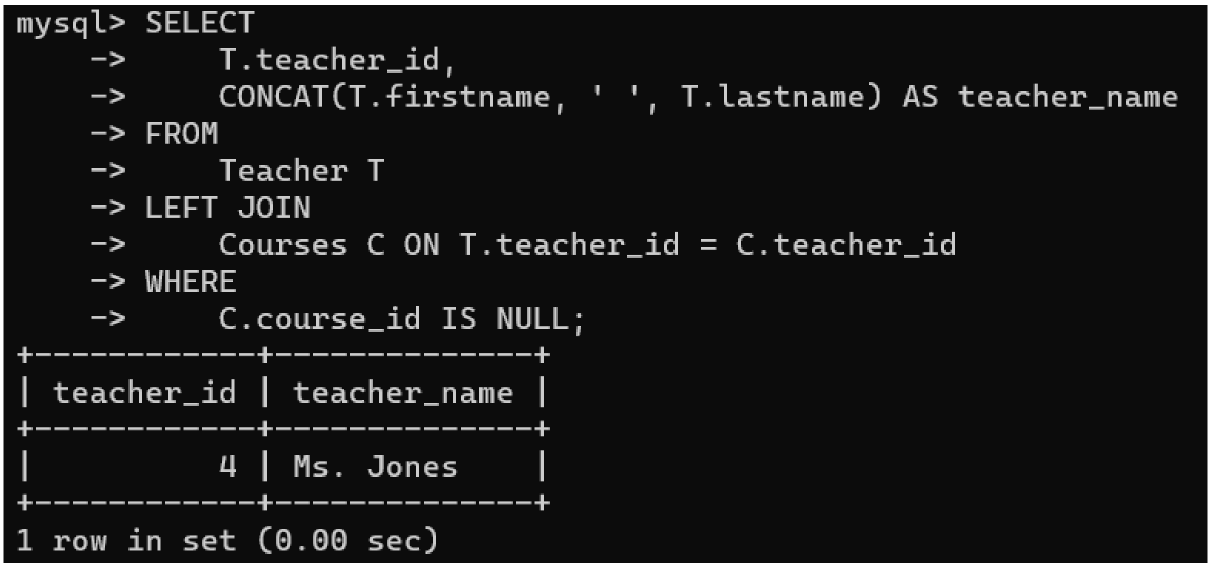
```
mysql> SELECT
    ->     C.course_id
    -> FROM
    ->     Courses C
    -> LEFT JOIN
    ->     Enrollments E ON C.course_id = E.course_id
    -> WHERE
    ->     E.enrollment_id IS NULL;
+-----------+
| course_id |
+-----------+
|         1 |
|         5 |
+-----------+
2 rows in set (0.00 sec)
```

9. Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.



```
mysql> SELECT
    ->     E1.student_id,
    ->     CONCAT(S.firstname, ' ', S.lastname) AS student_name,
    ->     COUNT(DISTINCT E1.course_id) AS enrolled_courses_count
    -> FROM
    ->     Enrollments E1
    -> JOIN
    ->     Students S ON E1.student_id = S.student_id
    -> GROUP BY
    ->     E1.student_id, S.firstname, S.lastname
    -> HAVING
    ->     COUNT(DISTINCT E1.course_id) > 1;
Empty set (0.00 sec)
```

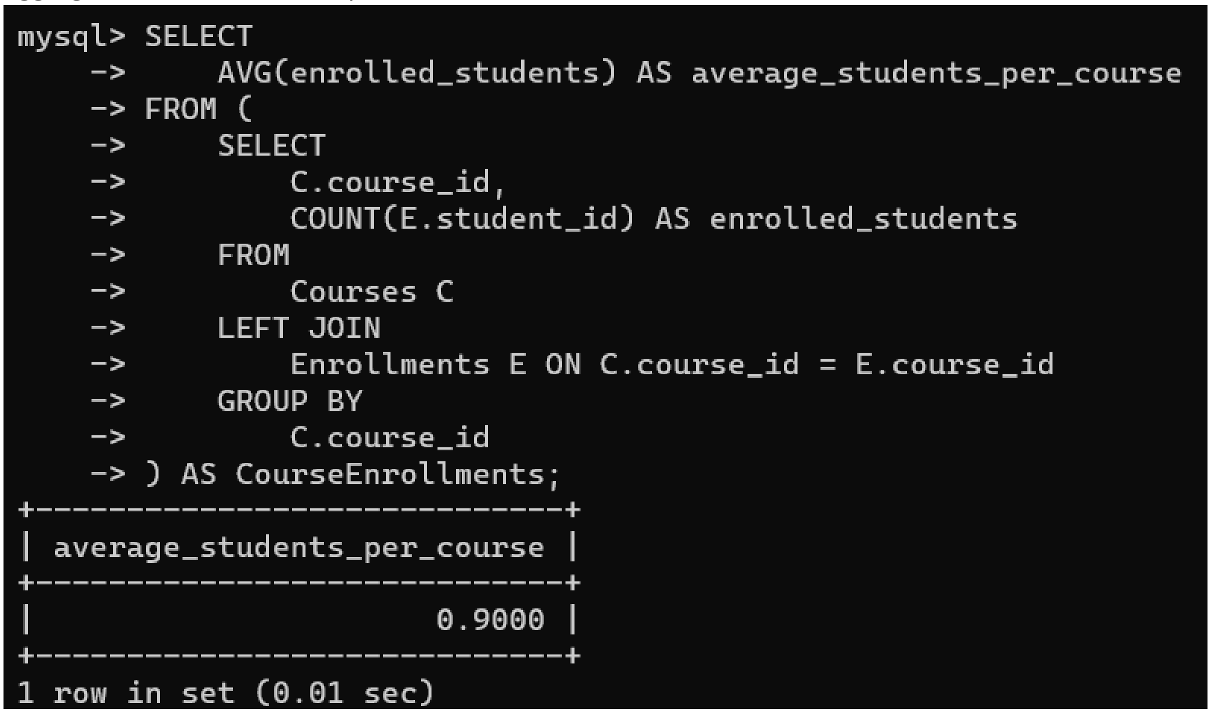
10. Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.



```
mysql> SELECT
    ->     T.teacher_id,
    ->     CONCAT(T.firstname, ' ', T.lastname) AS teacher_name
    -> FROM
    ->     Teacher T
    -> LEFT JOIN
    ->     Courses C ON T.teacher_id = C.teacher_id
    -> WHERE
    ->     C.course_id IS NULL;
+------------+--------------+
| teacher_id | teacher_name |
+------------+--------------+
|          4 | Ms. Jones    |
+------------+--------------+
1 row in set (0.00 sec)
```

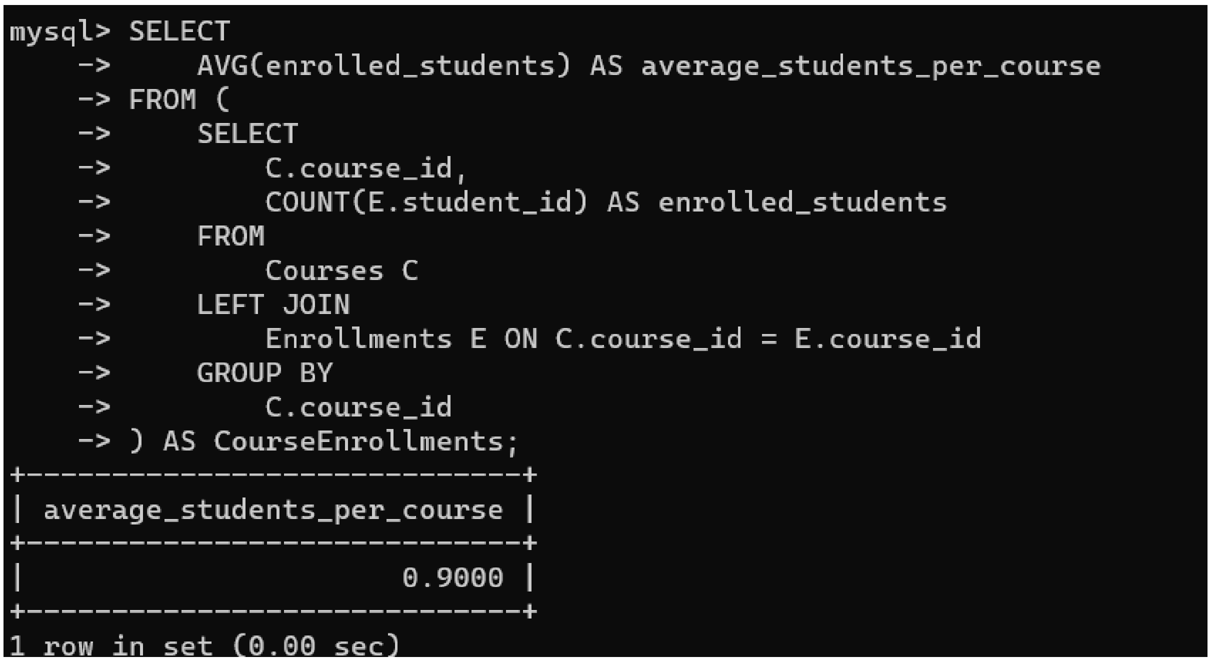
Task 4. Subquery and its type:

1. Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.



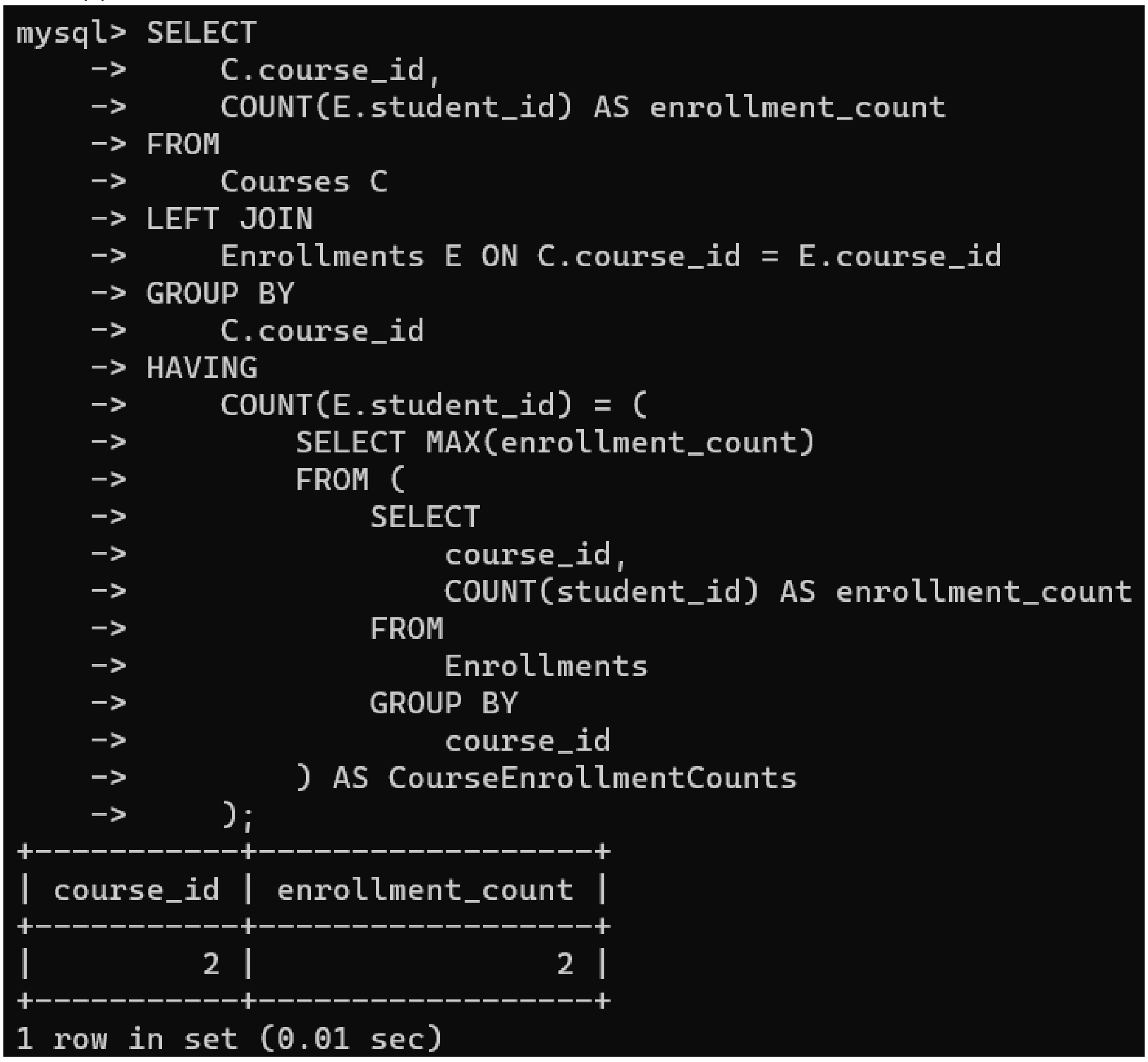
```
mysql> SELECT
    ->     AVG(enrolled_students) AS average_students_per_course
    -> FROM (
    ->     SELECT
    ->         C.course_id,
    ->         COUNT(E.student_id) AS enrolled_students
    ->     FROM
    ->         Courses C
    ->     LEFT JOIN
    ->         Enrollments E ON C.course_id = E.course_id
    ->     GROUP BY
    ->         C.course_id
    -> ) AS CourseEnrollments;
+-----------------------------+
| average_students_per_course |
+-----------------------------+
|                      0.9000 |
+-----------------------------+
1 row in set (0.01 sec)
```

2. Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.



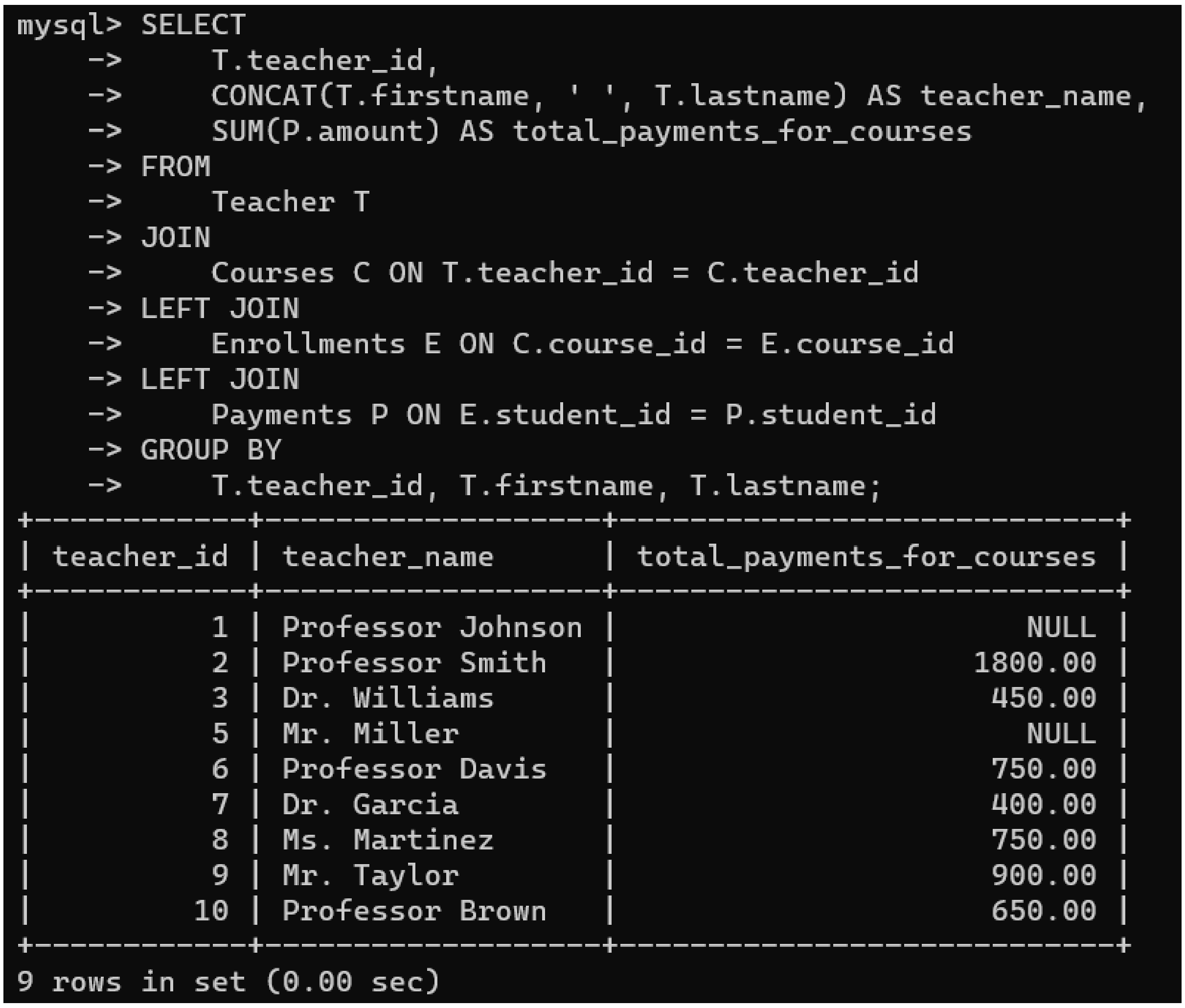
```
mysql> SELECT
    ->     AVG(enrolled_students) AS average_students_per_course
    -> FROM (
    ->     SELECT
    ->         C.course_id,
    ->         COUNT(E.student_id) AS enrolled_students
    ->     FROM
    ->         Courses C
    ->     LEFT JOIN
    ->         Enrollments E ON C.course_id = E.course_id
    ->     GROUP BY
    ->         C.course_id
    -> ) AS CourseEnrollments;
+-----------------------------+
| average_students_per_course |
+-----------------------------+
|                      0.9000 |
+-----------------------------+
1 row in set (0.00 sec)
```

3. Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.



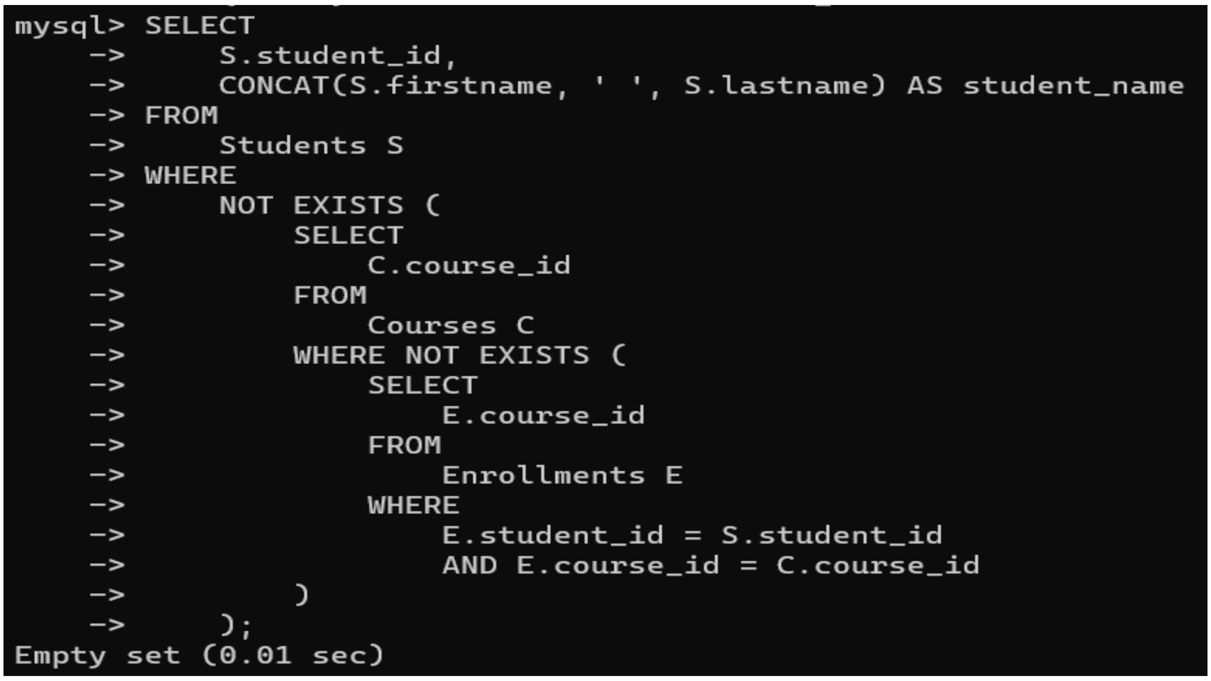
```
mysql> SELECT
    ->     C.course_id,
    ->     COUNT(E.student_id) AS enrollment_count
    -> FROM
    ->     Courses C
    -> LEFT JOIN
    ->     Enrollments E ON C.course_id = E.course_id
    -> GROUP BY
    ->     C.course_id
    -> HAVING
    ->     COUNT(E.student_id) = (
    ->         SELECT MAX(enrollment_count)
    ->         FROM (
    ->             SELECT
    ->                 course_id,
    ->                 COUNT(student_id) AS enrollment_count
    ->             FROM
    ->                 Enrollments
    ->             GROUP BY
    ->                 course_id
    ->         ) AS CourseEnrollmentCounts
    ->     );
+-----------+------------------+
| course_id | enrollment_count |
+-----------+------------------+
|         2 |                2 |
+-----------+------------------+
1 row in set (0.01 sec)
```

4. Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses



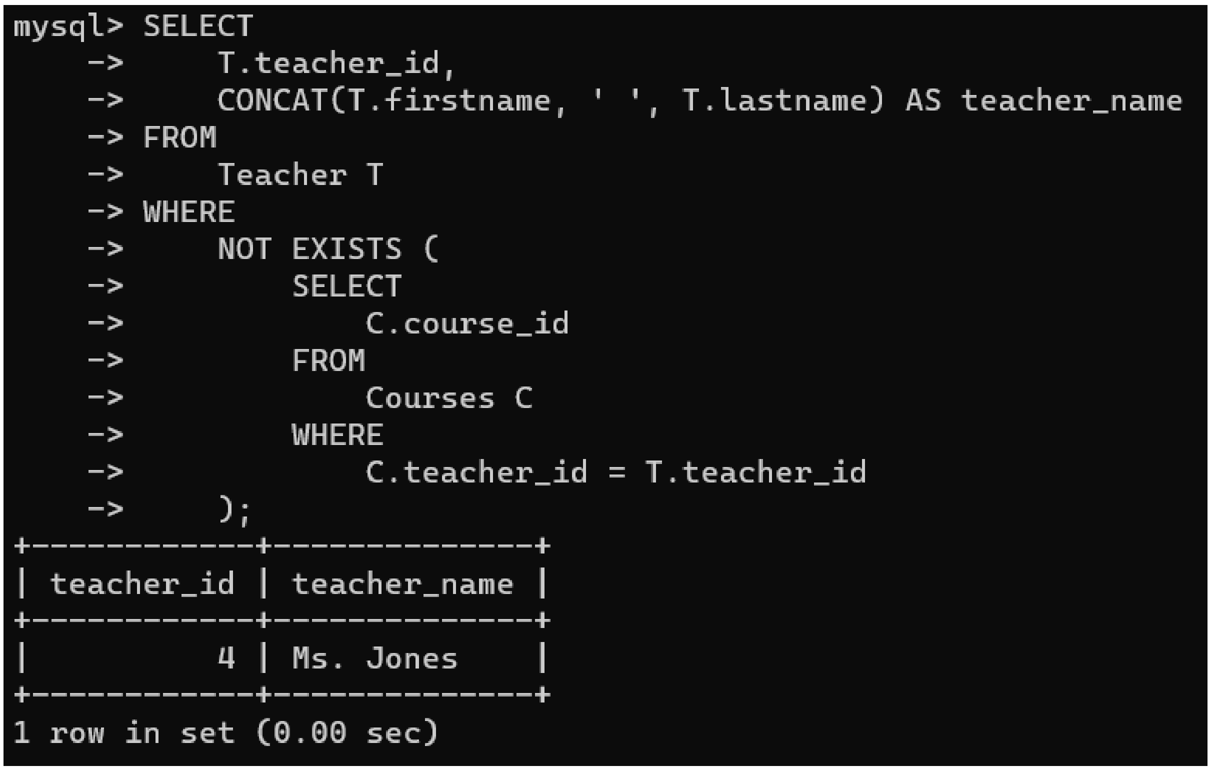
```
mysql> SELECT
    ->     T.teacher_id,
    ->     CONCAT(T.firstname, ' ', T.lastname) AS teacher_name,
    ->     SUM(P.amount) AS total_payments_for_courses
    -> FROM
    ->     Teacher T
    -> JOIN
    ->     Courses C ON T.teacher_id = C.teacher_id
    -> LEFT JOIN
    ->     Enrollments E ON C.course_id = E.course_id
    -> LEFT JOIN
    ->     Payments P ON E.student_id = P.student_id
    -> GROUP BY
    ->     T.teacher_id, T.firstname, T.lastname;
+------------+-------------------+----------------------------+
| teacher_id | teacher_name      | total_payments_for_courses |
+------------+-------------------+----------------------------+
|          1 | Professor Johnson |                       NULL |
|          2 | Professor Smith   |                    1800.00 |
|          3 | Dr. Williams      |                     450.00 |
|          5 | Mr. Miller        |                       NULL |
|          6 | Professor Davis   |                     750.00 |
|          7 | Dr. Garcia        |                     400.00 |
|          8 | Ms. Martinez      |                     750.00 |
|          9 | Mr. Taylor        |                     900.00 |
|         10 | Professor Brown   |                     650.00 |
+------------+-------------------+----------------------------+
9 rows in set (0.00 sec)
```

5. Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses



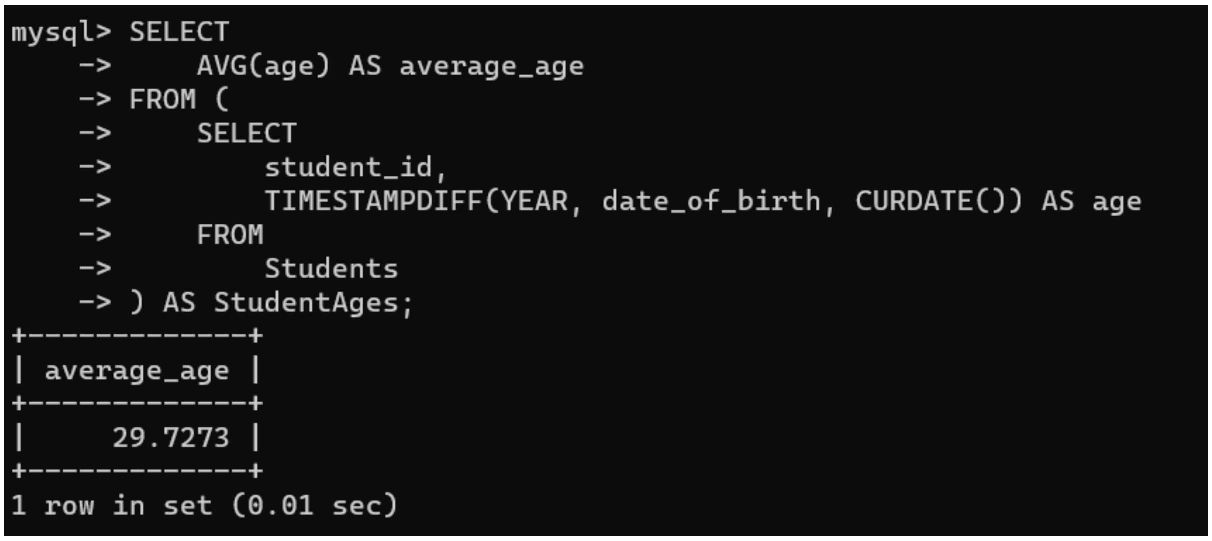
```
mysql> SELECT
    ->     S.student_id,
    ->     CONCAT(S.firstname, ' ', S.lastname) AS student_name
    -> FROM
    ->     Students S
    -> WHERE
    ->     NOT EXISTS (
    ->         SELECT
    ->             C.course_id
    ->         FROM
    ->             Courses C
    ->         WHERE NOT EXISTS (
    ->             SELECT
    ->                 E.course_id
    ->             FROM
    ->                 Enrollments E
    ->             WHERE
    ->                 E.student_id = S.student_id
    ->                 AND E.course_id = C.course_id
    ->         )
    ->     );
Empty set (0.01 sec)
```

6. Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.



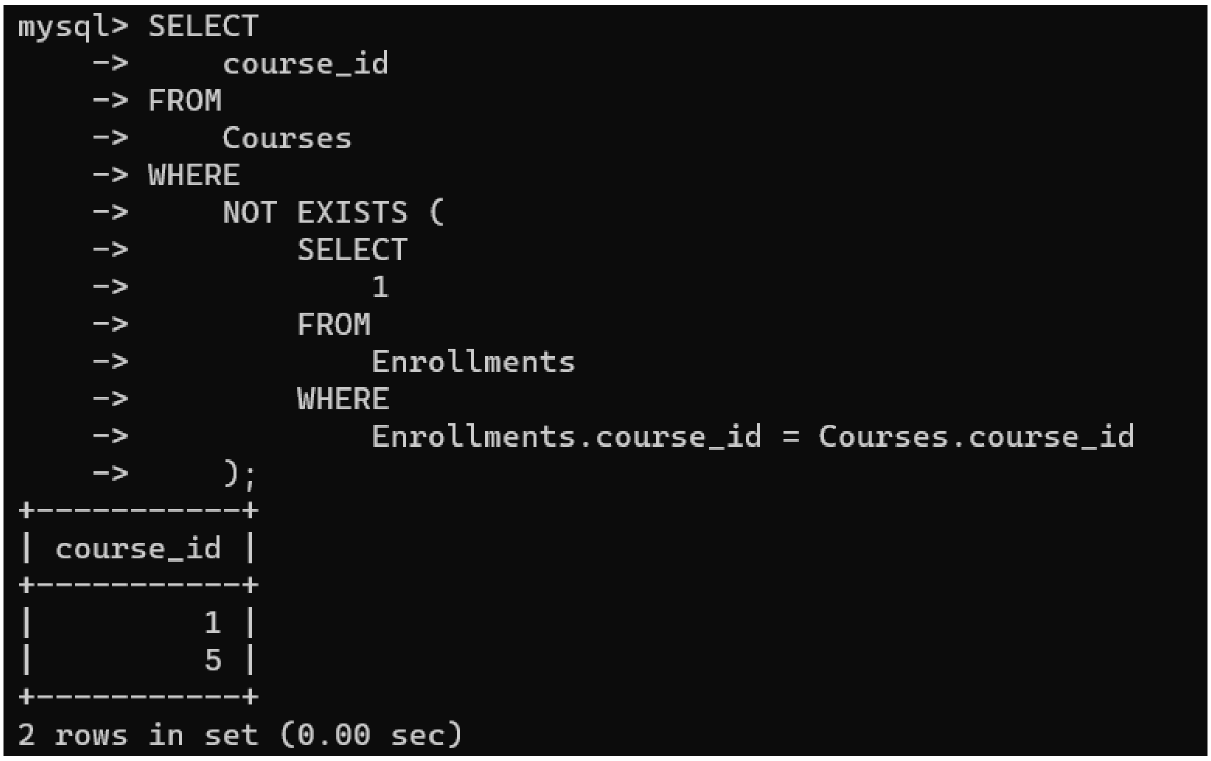
```
mysql> SELECT
    ->     T.teacher_id,
    ->     CONCAT(T.firstname, ' ', T.lastname) AS teacher_name
    -> FROM
    ->     Teacher T
    -> WHERE
    ->     NOT EXISTS (
    ->         SELECT
    ->             C.course_id
    ->         FROM
    ->             Courses C
    ->         WHERE
    ->             C.teacher_id = T.teacher_id
    ->     );
+------------+--------------+
| teacher_id | teacher_name |
+------------+--------------+
|          4 | Ms. Jones    |
+------------+--------------+
1 row in set (0.00 sec)
```

7. Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth



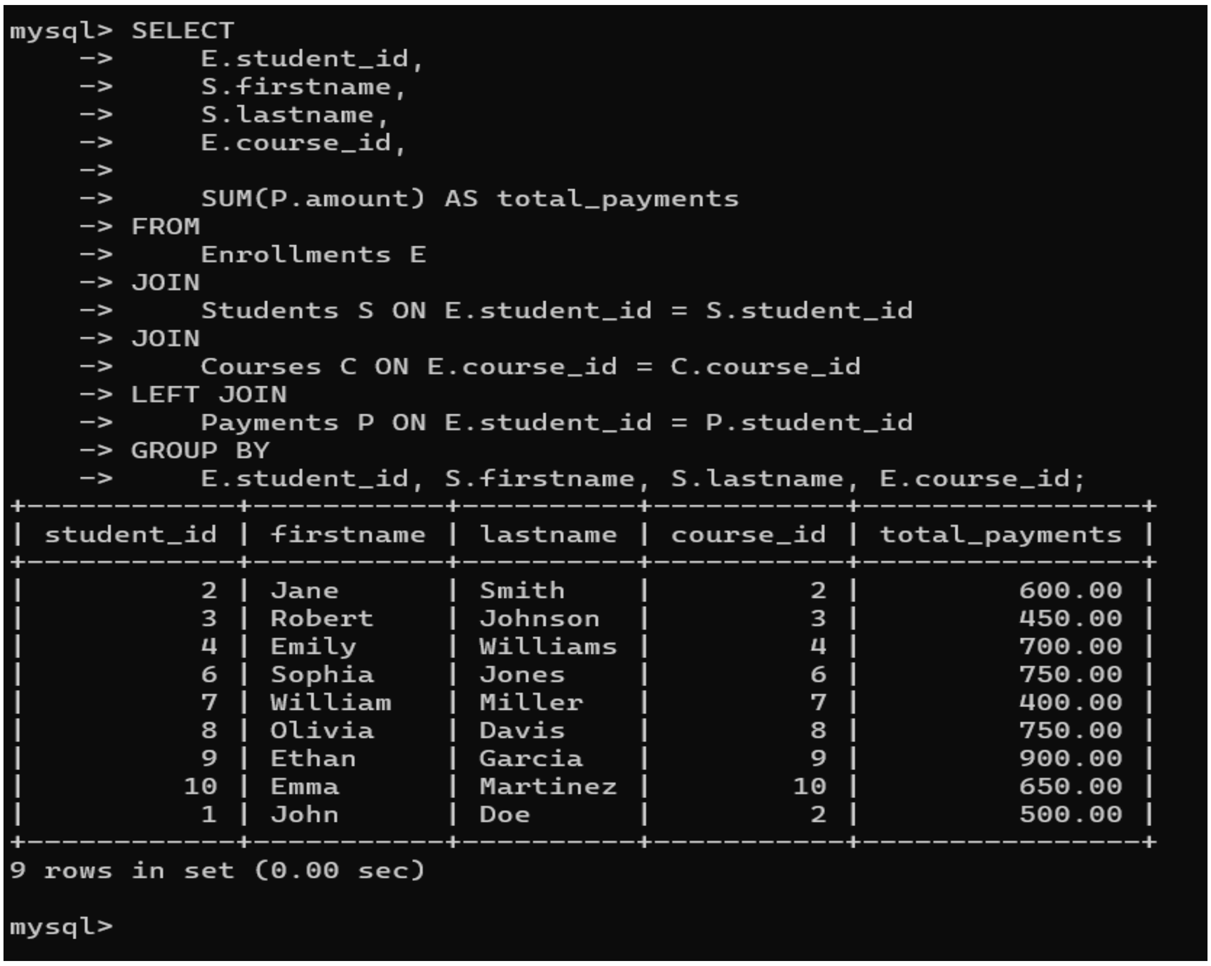
```
mysql> SELECT
    ->     AVG(age) AS average_age
    -> FROM (
    ->     SELECT
    ->         student_id,
    ->         TIMESTAMPDIFF(YEAR, date_of_birth, CURDATE()) AS age
    ->     FROM
    ->         Students
    -> ) AS StudentAges;
+-------------+
| average_age |
+-------------+
|     29.7273 |
+-------------+
1 row in set (0.01 sec)
```

8. Identify courses with no enrollments. Use subqueries to find courses without enrollment records.



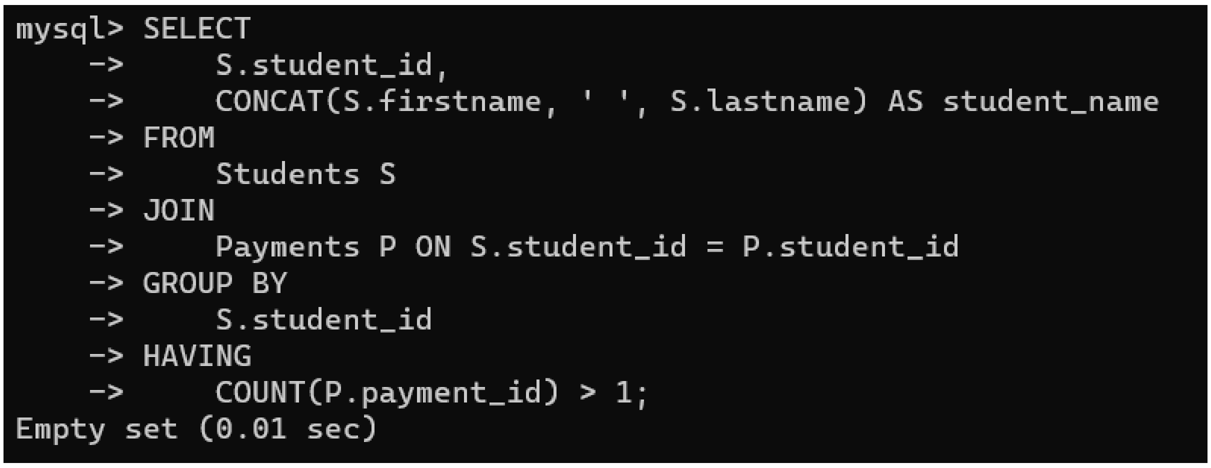
```
mysql> SELECT
    ->     course_id
    -> FROM
    ->     Courses
    -> WHERE
    ->     NOT EXISTS (
    ->         SELECT
    ->             1
    ->         FROM
    ->             Enrollments
    ->         WHERE
    ->             Enrollments.course_id = Courses.course_id
    ->     );
+-----------+
| course_id |
+-----------+
|         1 |
|         5 |
+-----------+
2 rows in set (0.00 sec)
```

9. Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.



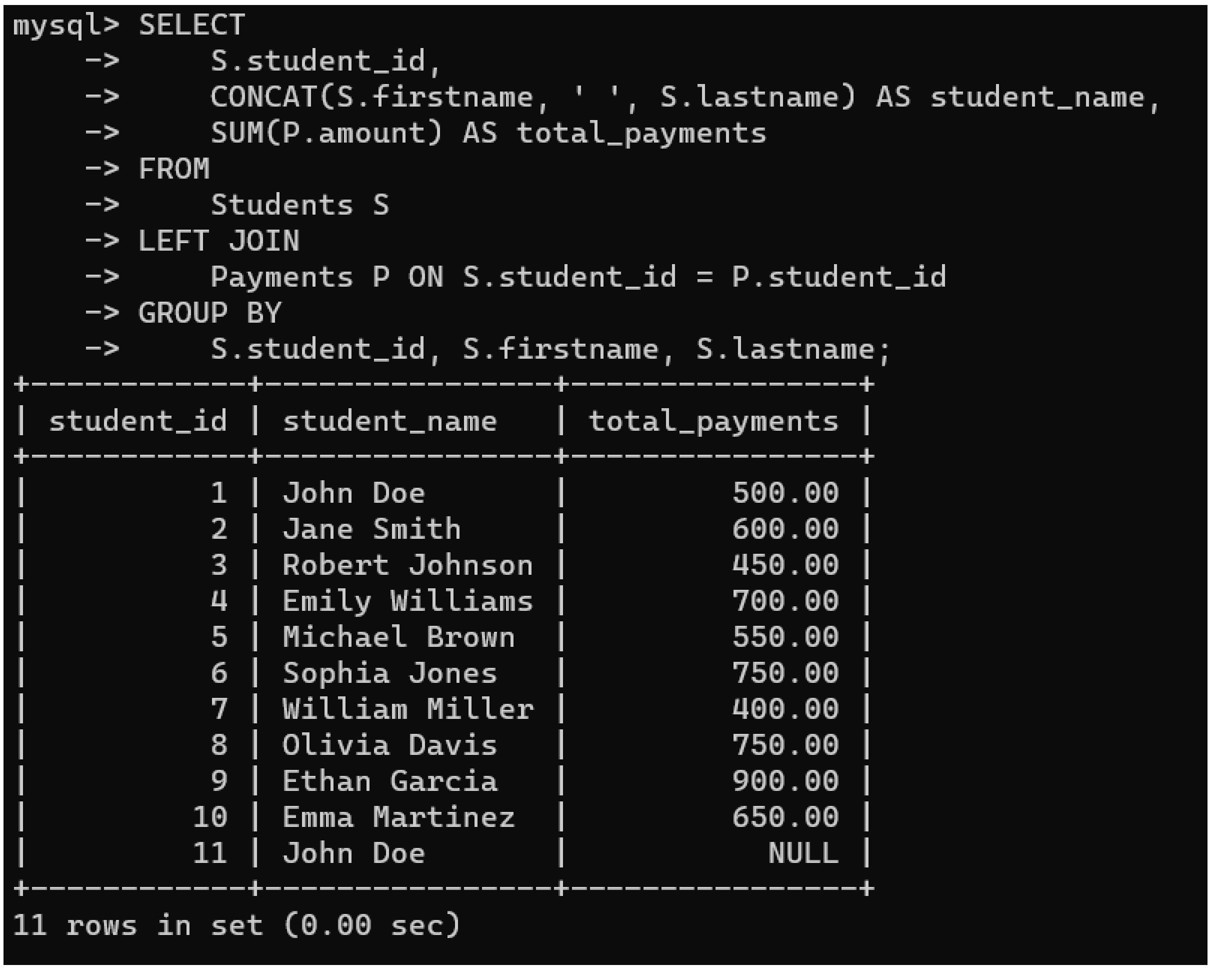
```
mysql> SELECT
    ->     E.student_id,
    ->     S.firstname,
    ->     S.lastname,
    ->     E.course_id,
    ->
    ->     SUM(P.amount) AS total_payments
    -> FROM
    ->     Enrollments E
    -> JOIN
    ->     Students S ON E.student_id = S.student_id
    -> JOIN
    ->     Courses C ON E.course_id = C.course_id
    -> LEFT JOIN
    ->     Payments P ON E.student_id = P.student_id
    -> GROUP BY
    ->     E.student_id, S.firstname, S.lastname, E.course_id;
+------------+-----------+----------+-----------+----------------+
| student_id | firstname | lastname | course_id | total_payments |
+------------+-----------+----------+-----------+----------------+
|          2 | Jane      | Smith    |         2 |         600.00 |
|          3 | Robert    | Johnson  |         3 |         450.00 |
|          4 | Emily     | Williams |         4 |         700.00 |
|          6 | Sophia    | Jones    |         6 |         750.00 |
|          7 | William   | Miller   |         7 |         400.00 |
|          8 | Olivia    | Davis    |         8 |         750.00 |
|          9 | Ethan     | Garcia   |         9 |         900.00 |
|         10 | Emma      | Martinez |        10 |         650.00 |
|          1 | John      | Doe      |         2 |         500.00 |
+------------+-----------+----------+-----------+----------------+
9 rows in set (0.00 sec)

mysql>
```

10. Identify students who have made more than one payment. Use subqueries and aggregate functions to count payments per student and filter for those with counts greater than one.



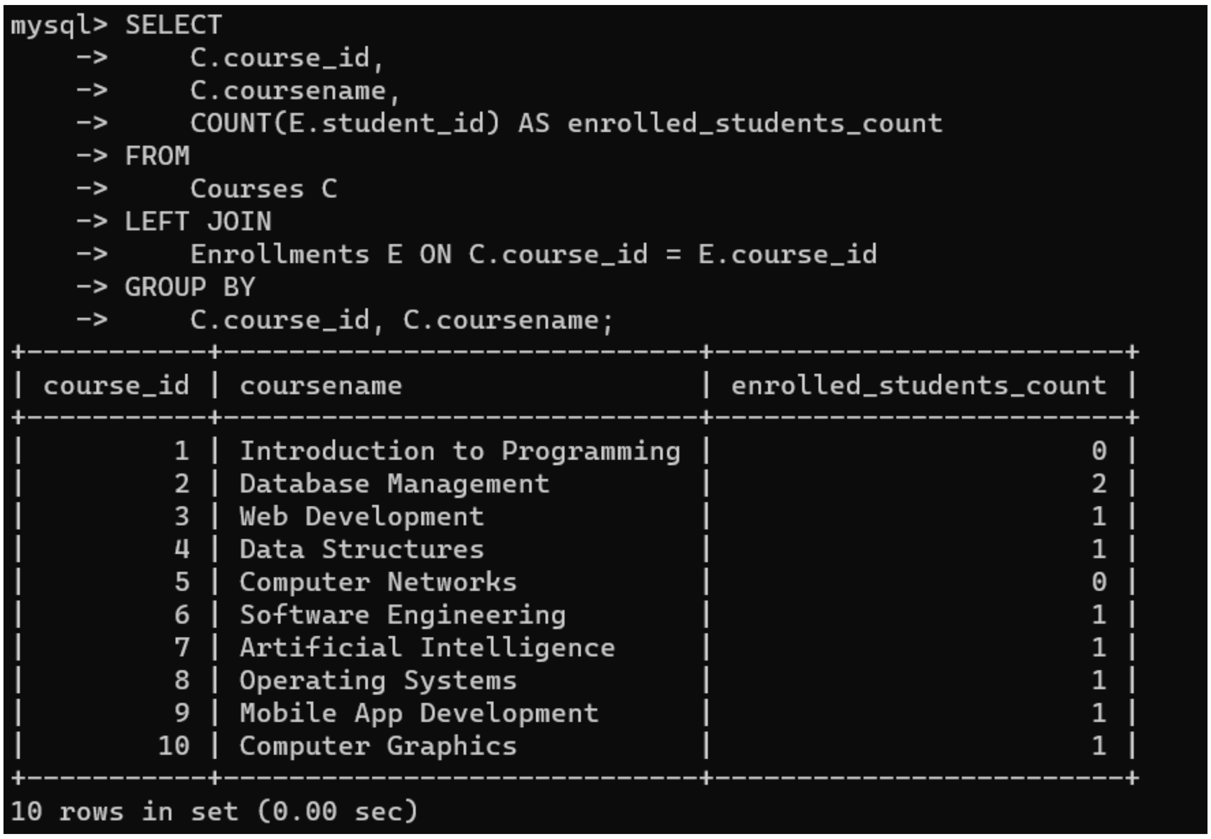
```
mysql> SELECT
    ->     S.student_id,
    ->     CONCAT(S.firstname, ' ', S.lastname) AS student_name
    -> FROM
    ->     Students S
    -> JOIN
    ->     Payments P ON S.student_id = P.student_id
    -> GROUP BY
    ->     S.student_id
    -> HAVING
    ->     COUNT(P.payment_id) > 1;
Empty set (0.01 sec)
```

11. Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.



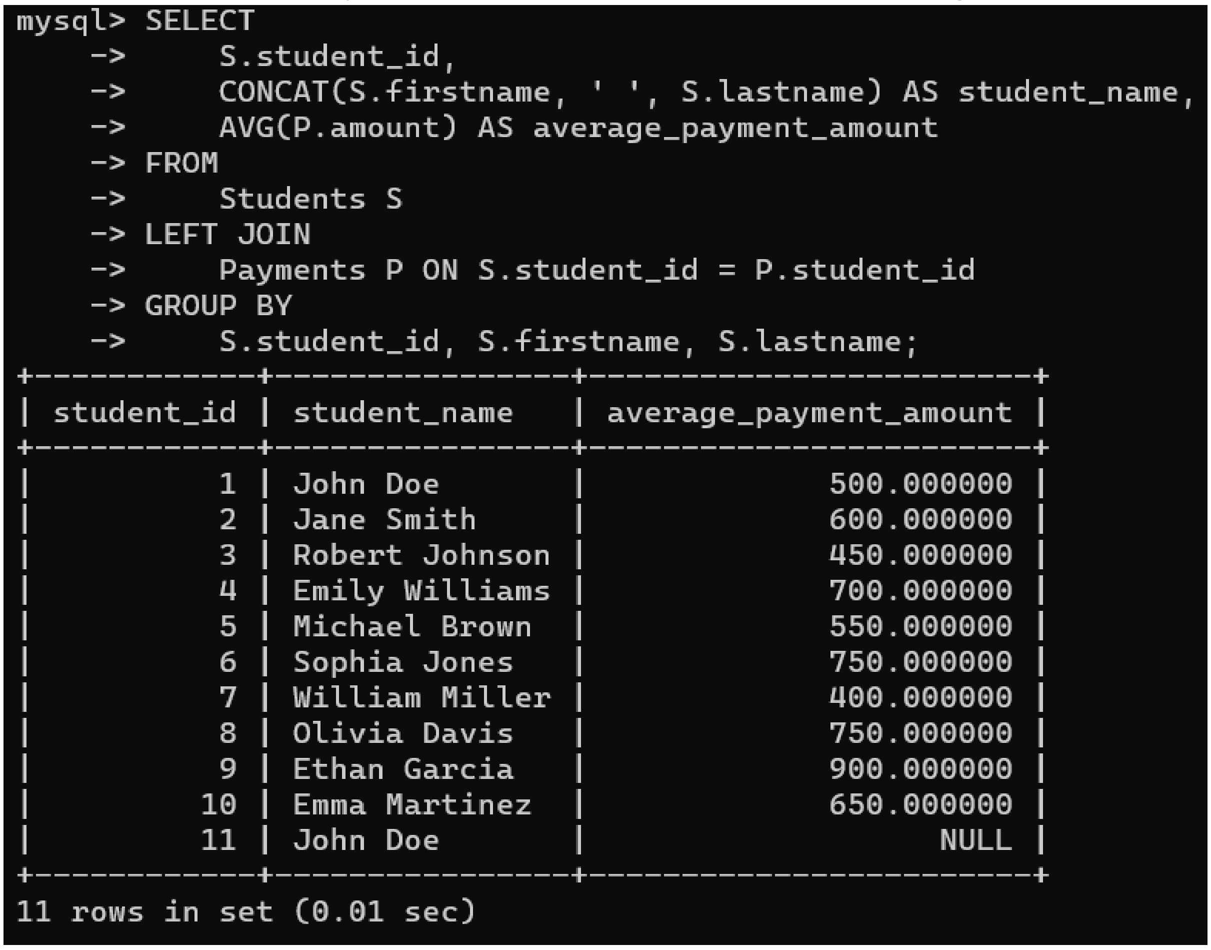
```
mysql> SELECT
    ->     S.student_id,
    ->     CONCAT(S.firstname, ' ', S.lastname) AS student_name,
    ->     SUM(P.amount) AS total_payments
    -> FROM
    ->     Students S
    -> LEFT JOIN
    ->     Payments P ON S.student_id = P.student_id
    -> GROUP BY
    ->     S.student_id, S.firstname, S.lastname;
+------------+----------------+----------------+
| student_id | student_name   | total_payments |
+------------+----------------+----------------+
|          1 | John Doe       |         500.00 |
|          2 | Jane Smith     |         600.00 |
|          3 | Robert Johnson |         450.00 |
|          4 | Emily Williams |         700.00 |
|          5 | Michael Brown  |         550.00 |
|          6 | Sophia Jones   |         750.00 |
|          7 | William Miller |         400.00 |
|          8 | Olivia Davis   |         750.00 |
|          9 | Ethan Garcia   |         900.00 |
|         10 | Emma Martinez  |         650.00 |
|         11 | John Doe       |           NULL |
+------------+----------------+----------------+
11 rows in set (0.00 sec)
```

12. Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.



```
mysql> SELECT
    ->     C.course_id,
    ->     C.coursename,
    ->     COUNT(E.student_id) AS enrolled_students_count
    -> FROM
    ->     Courses C
    -> LEFT JOIN
    ->     Enrollments E ON C.course_id = E.course_id
    -> GROUP BY
    ->     C.course_id, C.coursename;
+-----------+-----------------------------+-------------------------+
| course_id | coursename                  | enrolled_students_count |
+-----------+-----------------------------+-------------------------+
|         1 | Introduction to Programming |                       0 |
|         2 | Database Management         |                       2 |
|         3 | Web Development             |                       1 |
|         4 | Data Structures             |                       1 |
|         5 | Computer Networks           |                       0 |
|         6 | Software Engineering        |                       1 |
|         7 | Artificial Intelligence     |                       1 |
|         8 | Operating Systems           |                       1 |
|         9 | Mobile App Development      |                       1 |
|        10 | Computer Graphics           |                       1 |
+-----------+-----------------------------+-------------------------+
10 rows in set (0.00 sec)
```

13. Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.



```
mysql> SELECT
    ->     S.student_id,
    ->     CONCAT(S.firstname, ' ', S.lastname) AS student_name,
    ->     AVG(P.amount) AS average_payment_amount
    -> FROM
    ->     Students S
    -> LEFT JOIN
    ->     Payments P ON S.student_id = P.student_id
    -> GROUP BY
    ->     S.student_id, S.firstname, S.lastname;
+------------+----------------+------------------------+
| student_id | student_name   | average_payment_amount |
+------------+----------------+------------------------+
|          1 | John Doe       |             500.000000 |
|          2 | Jane Smith     |             600.000000 |
|          3 | Robert Johnson |             450.000000 |
|          4 | Emily Williams |             700.000000 |
|          5 | Michael Brown  |             550.000000 |
|          6 | Sophia Jones   |             750.000000 |
|          7 | William Miller |             400.000000 |
|          8 | Olivia Davis   |             750.000000 |
|          9 | Ethan Garcia   |             900.000000 |
|         10 | Emma Martinez  |             650.000000 |
|         11 | John Doe       |                   NULL |
+------------+----------------+------------------------+
11 rows in set (0.01 sec)
```